Tutorial y manual de Entidad 3D v3.7d

por Jordi Pérez (C) Enero / 2007

Version Imprimible No oficial de Entidad 3Dv3.7d

por Jaime Yemaíl Enero / 2007

Pasos del tutorial:
1º- Introducción: **4**

2º- Nombres de ficheros: **5**

3º- Ficheros 3D .act: **8**

4º- Objetos: **9**

5º- Abriendo puertas: **11**

6º- Personajes: **14**

7º- Enemigos: **15**

Entidades disponibles: **16**

DeathMatchStart (inicio del jugador) 16

Command (comandos en escenario) 19

CommandModel (comandos en modelo) 20

SpecificActor (personajes secundarios) 21

BotActorStart (enemigos/personajes móviles) 23

Door (puertas) 29

MovingPlat (ascensores) 30

LuzJugador (luz del jugador) 31

Configuracion (configuración general) 32

ConfigEnemigos (configuración general de enemigos): 35

VerArma (arma en primera persona) 36

Explosion (explosiones) 36

ElectricBolt (sonidos 3D posicionales) 37

ItemObject (objetos) 38

ItemVida (vida extra) 41

ItemHealth (salud) 41

ItemArmor (armadura) 42

ItemGrenade (arma 2 Lanza-granadas) 43

ItemRocket (arma 3 Lanza-cohetes) 43

ItemShredder (arma 4 Fusil) 43

ItemBlasterAmmo (cargador arma 1) 44

ItemGrenadeAmmo (cargador arma 2) 44

ItemRocketAmmo (cargador arma 3) 44

ItemShredderAmmo (cargador arma 4) 45

ModelController (animación de bloques) 45

CamaraFija (cámaras fijas en el escenario) 46

Animacion (animación en el escenario) 49

DynamicLight (luces dinámicas) 53

Corona (corona de luz) 55

ConfigArmaUno (configuración arma 1) 56

ConfigArmaDos (configuración arma 2) 60

ConfigArmaTres (configuración arma 3) 63

ConfigArmaCuatro (configuración arma 4) 66

ConfigSonidosEne (configuración sonidos enemigos) 69

ConfigSonidosJug (configuración sonidos jugador) 71

Objetivo (objetivos de la misión) 72

SunLight (luz de Sol) 73

Posicion (rutas del enemigo) 73

ChangeLevel (cargar nivel) 74

Fx_Surtidor (Partículas surtidor) 74

Fx_Lluvia (Partículas lluvia) 76

Fx_Sprite (Partículas sprites) 78

Fx_Niebla (Niebla en la distancia) 79

Fx_ArmaEnergia (Armas de energía) 80

TeclaComandos (configurar teclas) 82

Marcador (configurar marcadores) 82

MarcadorBarra (configurar barras de marcador) 84

AccionMenu (configurar menú de acciones) 87

VolarJugador (modo volar del jugador) 89

Comandos disponibles: **90**

SONF (Activar música de fondo) 90

SONP (Pausa en música de fondo) 90

SONR (Reanudar música de fondo) 90

SONI (Efecto de sonido) 90

SON0 (Detiene sonidos 3D) 90

SON1 (Reanuda sonidos 3D) 90

SODF SODP SODR (Efecto de sonido continuo de fondo) 91

TEXT (Pone texto en pantalla) 91

DIAL (Pone un diálogo en pantalla) 92

QINV (Elimina objeto del inventario) 92

QTIN (Elimina objetos del inventario) 92

PINV (Pone objeto en el inventario) 93

QINVUSO (Resta usos de objeto del inventario) 93

QINVVENDER (Vende objeto del inventario) 94

PONE (Asigna un valor a una variable) 94

PONA (Asigna un valor al azar a una variable) 94

OPE+ (Suma en variable) 95

OPE- (Resta en variable) 95

OPE* (Multiplicación en variable) 95

OPE/ (División en variable) 95

OPEC (Añadir texto a una variable) 96

IF ELSE ENDIF (Condiciones) 96

EXIT (Finalizar proceso de comandos) 98

VIS0 (Hace invisible un personaje) 99

VIS1 (Hace visible un personaje) 99

MOTC (Cambia la animación cíclica de un personaje) 99

MOTI (Cambia la animación de un personaje) 100

PERS (Cambia el personaje protagonista) 100

OPCI (Plantea opciones al jugador) 101

IMG1 (Superpone imágenes 2D 8bits) 102

IMG0 (Hace desaparecer imágenes 2D 8bits superpuestas) 105

OVERLAY1 OVERLAY0 (Superpone imágenes 2D 24bits) 106

MSJ1 MSJ0 (Presenta textos o números en pantalla) 109

TEM1 (Activa temporizadores) 113

TEM0 (Desactiva temporizadores) 114

EXPL (Produce explosiones) 115

BLQ0 (Detiene animación bloque) 115

BLQ1 (Reanuda animación bloque) 115

QVID (Quita una vida al jugador) 116

DVID (Da una vida al jugador) 116

FINJ (Fin del juego) 117

CAM1 (Activa cámara fija) 117

CAM0 (Desactiva cámara fija) 117

CHKP (Checkpoint) 118

LAT0 LAT1 (Bloqueo movimiento lateral) 118

CTEX (Cambio de textura) 119

ANI1 ANI0 (Control animaciones) 120

LUZ1 (Activa una luz) 120

LUZ0 (Desactiva una luz) 120

COR1 (Activa una luz corona) 120

COR0 (Desactiva una luz corona) 121

TEA1 (Activa una textura animada) 121

TEA0 (Desactiva una textura animada) 121

QSAL (Quita salud) 121

DSAL (Incrementa salud) 121

MOV1 MOV0 (Inmoviliza al jugador) 122

TELETRANSPORTE (Teletransporta al jugador) 122

PAN1 (Añade objetivo) 122

PAN0 (Oculta objetivo) 122

OBJ1 (Objetivo cumplido) 123

INPU (Entrar texto) 123

NIE0 (Desactiva niebla de fondo) 124

NIE1 (Activa niebla de fondo) 124

ARMAENERGIA (Activa arma de energía) 125

SALTO (Altura del salto del jugador) 126

VELOCIDAD (Velocidad del jugador) 126

SALIRJUEGO (Salir del juego) 126

EXEC (Lanzar un programa externo) 126

BARRAACTIVAR (Activar barra de marcador) 127

BARRADESACTIVAR (Desactivar barra de marcador) 127

BARRAVISIBLE (Barra de marcador visible) 128

BARRAINVISIBLE (Barra de marcador invisible) 128

BARRAVALOR (Valor de barra de marcador) 128

SETCLAVEJUEGO (Estableca clave del juego) 128

CHKCLAVEJUEGO (Chequea clave del juego) 129

ENERGIAVOLAR (Incrementa energía de vuelo) 129

JUGADORVOLAR1 (Activar volar para el jugador) 130

JUGADORVOLAR0 (Desactivar volar para el jugador) 130

SALVAVARIABLE (Salva una variable en un fichero) 130

CARGAVARIABLE (Carga una variable de un fichero) 131

Carpetas, ficheros de imágenes 2D y sonidos: **132**
Imágenes de introducción 132
Imagen inicio nivel 133
Imagen cargando nivel 133
Imagen fin de partida 133
Imágenes del menú 134
Imágenes de los textos 136
Imágenes del marcador 136
Imagen de la consola 137
Imagen de objetos del inventario 138
Música de introducción 139
Música de fondo 139
Música fin de partida 140
Otros ficheros 140

Anexos: **141**
Inventario de objetos y menú de acciones 141

Puntuación 142

Dinero 142

Cargar partida 143

Texturas animadas 143

Cámaras en Entidad 3D 145

Encriptación de datos 151

Generación de mazmorras 153

Ficheros .LOG y .INI: **158**
'**Informe_Entidad_3D.Log** 158
(Seguimiento de procesos)'

Entidad3D.ini 158
(Configuración de Entidad 3D)

D3D24.ini 159
(Configuración Direct 3D, OpenGL)

Mazmorra.ini 160
(Configuración del generador de mazmorras y laberintos)

El juego de ejemplo 'Secta' 160
Modificación del juego

Controles del juego: 160
Teclas del juego y ratón 160
Joystick o Gamepad 161

Funcionamiento del juego fuera de 'World Editor': **162**
Preparando un acceso directo 162
Empaquetado del juego en un archivo .ZIP 163

Programas que incluye la última distribución de Entidad 3D: **164**
World Editor 1.0
World Editor 2.0
World Editor Pro 164
Texture Packer 165
Actor Viewer 165
Entidad 3D Actor Viewer 166
Actor Studio 166
Skin replacer 167
Dungeon Maker 167
Entidad 3D 168

Historial de versiones: **168**

Entidad 3D v1.1 169
(Distribuido el 23/Marzo/2004)

En **Entidad 3D v2.0**
(Nuevas características con respecto a Entidad 3D v1.1)

En **Entidad 3D v2.1** 175
(Nuevas características con respecto a Entidad 3D v2.0)

En **Entidad 3D v2.2** 177
(Nuevas características con respecto a Entidad 3D v2.1)

En **Entidad 3D v2.3a** 178
(Nuevas características con respecto a Entidad 3D v2.2)

En **Entidad 3D v2.4** 178
(Nuevas características con respecto a Entidad 3D v2.3a)

En **Entidad 3D v3.0** 181
(Nuevas características con respecto a Entidad 3D v2.4)

En **Entidad 3D v3.2** 186
(Nuevas características con respecto a Entidad 3D v3.0)

En **Entidad 3D v3.3** 188
(Nuevas características con respecto a Entidad 3D v3.2)

En **Entidad 3D v3.4** 190
(Nuevas características con respecto a Entidad 3D v3.3)

En **Entidad 3D v3.4c** 192
(Nuevas características con respecto a Entidad 3D v3.4)

En **Entidad 3D v3.5a** 193
(Nuevas características con respecto a Entidad 3D v3.4c)

En **Entidad 3D v3.6a** 193
(Nuevas características con respecto a Entidad 3D v3.5a)

En **Entidad 3D v3.6b** 195
(Nuevas características con respecto a Entidad 3D v3.6a)

En **Entidad 3D v3.7a** 195
(Nuevas características con respecto a Entidad 3D v3.6b)

En **Entidad 3D v3.7d** 196
(Nuevas características con respecto a Entidad 3D v3.7a)

Niveles anteriores: **196**
Utilización de niveles de anteriores versiones de Entidad 3D

Algunas cuestiones: **196**
Preguntas y respuestas más frecuentes

Licencias: **199**
Licencia de uso de Entidad 3D (Freeware) 199
Licencia de uso del juego 'Secta' (Freeware) 200
Licencia comercial 201
Genesis3D SDK 202
Librería Bass 205
Librería FreeImage 206

Descarga este tutorial/manual en su revisión 3.7a (14 Mbytes) 210

Revisión de este manual: 3.7d **210**

1º- Introducción al tutorial de 'Entidad 3D'

Este es un tutorial sobre el funcionamiento de 'Entidad 3D'. Programa que nos permitirá hacer realidad nuestro juego como tal. En realidad este programa, cuyo ejecutable es 'entidad3d.exe' ya es nuestro juego, sólo que tendremos que pasarle una serie de parámetros para indicarle el escenario a cargar, y dentro de este escenario especificaremos el resto de elementos que formarán nuestro juego, como personajes, objetos y desarrollo del mismo.
En este tutorial no se incluye ninguna información referente a como diseñar objetos o personajes 3D, texturas, sonidos o músicas. 'Entidad 3D' no hace eso (para eso ya hay herramientas específicas), 'Entidad 3D' es sólo una herramienta más de todo el proceso de desarrollo de un juego y es la que gestionará todos los recursos gráficos y sonoros para formar un juego, es básicamente el programa del juego ... así que eso que te ahorrarás de programar.
Si no quieres o no sabes crear objetos 3D, gráficos o sonidos, entonces tendrás que utilizar todo lo que encuentres por Internet o lo que te pueda servir de esta distribución de 'Entidad 3D', por lo demás con 'Entidad 3D' ya tienes todo lo necesario para poder crear un juego 3D con desarrollo y distribuible en un solo fichero de forma independiente.
Es imprescindible antes de comenzar este tutorial que te hayas leído previamente el tutorial de 'World Editor' para entender muchos conceptos de este tutorial, ya que básicamente seguiremos utilizando 'World Editor' pero con las nuevas entidades particulares de 'Entidad 3D' para desarrollar toda la interactividad de nuestro juego.
Es recomendable que comiences con los pasos numerados de la sección 'Pasos del tutorial' y en ese orden, ya que se trata de un proceso de aprendizaje progresivo.

**/// (C) 2006 por Jordi Pérez **

2º- Nombres de ficheros

Con la finalidad de que puedan coexistir varios juegos distintos al mismo tiempo en Entidad 3D, pero utilizando las mismas carpetas, se ha establecido una serie de nombres de ficheros para que Entidad 3D los cargue en función del nivel que se está jugando, de esta forma podremos diferenciar diversos ficheros de audio, imágenes y personajes para cada juego o nivel en concreto.
Por este motivo es importante que establezcas los nombres de ficheros que se indican en este manual para que Entidad 3D sea capaz de encontrarlos y establecerlos para el juego o nivel correspondiente.
Por ejemplo: Tenemos un juego que consta de tres niveles que se cargarán por este orden a medida que el jugador avance en su desarrollo:

puente.bsp
carretera.bsp
escape.bsp

Cuando este manual se refiera a ficheros que afectan a todo el juego, te indicará que pongas delante del fichero el nombre del primer nivel de tu juego. Se indicará de esta forma:

*PrimerNivel*_intro.mp3

Con lo que para nuestro ejemplo el nombre real del fichero tendría que ser:

puente_intro.mp3

Y cuando se refiera a crear ficheros particulares para cada nivel, lo encontrarás indicado de esta forma:

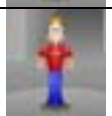
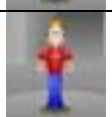
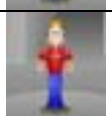
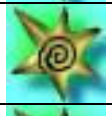
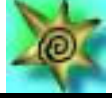
*Nivel*_portada.bmp

Con lo que por ejemplo para el nivel 'carretera' el nombre real sería :

carretera_portada.bmp

De todas formas si creas los niveles sin estos nombres particulares, Entidad 3D utilizará unos ficheros por defecto o simplemente si no los encuentra no los utilizará, el caso es que el programa no producirá ningún error y podrás probar tu nivel sin problemas.

Tabla de ejemplo:
Para que tengas una visión general de este sistema de ficheros, a continuación se muestra una tabla de los ficheros que existirían para el juego del ejemplo anterior que recordemos consta de tres niveles llamados 'puente' 'carretera' y 'escape':

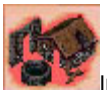
Tipo de fichero	Nombre de fichero	Función del fichero	General / Particular
	puente_intro1.bmp	Primera imagen de la introducción del juego	
	puente_intro2.bmp	Segunda imagen de la introducción del juego	
	puente_intro3.bmp	Tercera imagen de la introducción del juego	
	puente_intro.mp3	Fondo musical para la introducción del juego	
	puente_final.mp3	Música de final de partida	
	puente_player1.act	Personaje protagonista inicial	
	puente_player2.act	Personaje protagonista alternativo	
	puente_enemigo1.act	Personaje enemigo número 1	
	**puente_enemigo2.act**	Personaje enemigo número 2	
	puente_enemigo3.act	Personaje enemigo número 3	
	puente_enemigo4.act	Personaje enemigo número 4	
	puente_enemigo5.act	Personaje enemigo número 5	
	**puente_enemigo6.act**	Personaje enemigo número 6	
	puente_enemigo7.act	Personaje enemigo número 7	
	**puente_enemigo8.act**	Personaje enemigo número 8	
	puente_creditos.bmp	Títulos de crédito del juego	
	**puente_hud_graficos.bmp**	Gráfico 2D fin de partida y punto de mira	
	**puente_graficos_superpuestos.bmp**	Gráficos 2D superpuestos	

	puente_objetos_distintos.bmp	Iconos 2D de objetos	
	puente_final_juego.bmp	Imagen de final de juego	
	puente.bsp	Primer nivel del juego	
	puente_portada.bmp	Imagen inicial justo antes de cargar el nivel 'puente'	
	puente_cargando.bmp	Imagen mientras se está cargando el nivel 'puente'	
	puente_fondo.mod	Fondo musical mientras se juega al nivel 'puente'	
	carretera.bsp	Segundo nivel del juego	
	carretera_portada.bmp	Imagen inicial justo antes de cargar el nivel 'carretera'	
	carretera_cargando.bmp	Imagen mientras se está cargando el nivel 'carretera'	
	carretera_fondo.s3m	Fondo musical mientras se juega al nivel 'carretera'	
	escape.bsp	Tercer nivel del juego	
	escape_portada.bmp	Imagen inicial justo antes de cargar el nivel 'escape'	
	escape_cargando.bmp	Imagen mientras se está cargando el nivel 'escape'	
	escape_fondo.wav	Fondo musical mientras se juega al nivel 'escape'	

Indica que es un fichero de imagen. Sólo se permiten en formato .BMP

Indica que es un fichero de sonido en cualquiera de los siguientes formatos: mp3, wav, mod, s3m, xm, it, mtm, mo3, mp1, mp2, ogg

Indica que es un fichero de personaje en formato .ACT

 Indica que es un fichero de nivel en formato .BSP de World Editor

Indica que es un fichero genérico para todo el juego

Indica que es un fichero particular para un nivel en concreto

*Los ficheros .BMP tienen que estar en la carpeta 'bmp' del entorno de desarrollo de EntidaD 3D, por defecto en **'C:\entidad_3d\bmp\'**, menos el fichero 'PrimerNivel_creditos.bmp' que ha de estar en **'C:\entidad_3d\menu\bmp\'**.*
*Los ficheros de sonido tienen que estar en la carpeta 'wav' del entorno de desarrollo de EntidaD 3D, por defecto en **'C:\entidad_3d\wav\'**.*
*Los ficheros .ACT tienen que estar en la carpeta 'actors' del entorno de desarrollo de EntidaD 3D, por defecto en **'C:\entidad_3d\actors\'**.*

3º- Ficheros 3D .act:

Voy a comenzar explicando el formato de los ficheros de modelos 3D que utiliza 'Entidad 3D', ya que son la base fundamental para definir objetos y personajes en nuestro juego.
Para representar los objetos 3D en nuestro juego, 'Entidad 3D' utiliza los ficheros con extensión .act tanto para objetos como para personajes o enemigos. Este tipo de formato de modelos 3D no es un formato muy extendido y se usa particularmente en programas basados en el motor 3D 'Genesis3D', como es el caso de 'Entidad 3D' o del programa 'Reality Factory'.
Existe un conversor de varios formatos 3D al formato .act llamado 'MilkShape 3D', pero es un programa Shareware que no se incluye en 'Entidad 3D'. Esta utilidad puede trabajar con modelos de juegos como Half-Life, Quake 2, Quake III, Unreal, Serious Sam, Max Payne, The Sims, etc...
Utilizando el tutorial para convertir personajes de Half-Life a personajes de Entidad 3D que podrás encontrar en la página de 'Entidad 3D', podrás convertir diversos personajes gratuitos del formato Half-Life (ficheros .mdl) al formato .act con excelentes resultados.
Pero si no quieres utilizar 'MilkShape 3D', tendrás que utilizar los ficheros .act que puedas encontrar en Internet o los del propio 'Entidad 3D'. De todas formas, en la web de 'Entidad 3D' podrás encontrar periódicamente diversos modelos .act que yo mismo iré convirtiendo de modelos de 'Half-Life'.
En 'Entidad 3D' estos ficheros .act los podrás encontrar en dos lugares: en la carpeta 'Actors' y en la carpeta 'Objects'. Puedes visualizarlos con la utilidad 'Actor Viewer' que se te instaló en tu sistema cuando instalaste 'Entidad 3D'.

Actor Viewer

Los ficheros .act contienen básicamente el objeto o personaje 3D junto con una serie de animaciones predefinidas. Cada animación tiene su propio nombre, nombres que pueden variar entre un fichero .act y otro. Cada archivo .act puede tener un número variable de animaciones, incluso ninguna en el caso de un objeto estático. Así pues nos podemos encontrar con un fichero 'player1.act' con las siguientes animaciones:

Animación 'Idle'

Animación 'Shoot'

Animación 'Run'

En el caso de los .act utilizados para personajes en el juego de ejemplo 'Secta', la mayoría constan de 40 animaciones distintas, aunque eso no quiere decir que se utilicen todas en el juego.
El nombre de las animaciones puede ser distinto en función del autor de los ficheros .act, para saber los nombres de las animaciones tendremos que utilizar 'Actor Viewer'.
Una cosa **MUY IMPORTANTE** en 'Entidad 3D' es que al menos 9 de esos nombres de animaciones han de tener un nombre exacto, ya que 'Entidad 3D' los busca de forma automática para animar al personaje protagonista y a los enemigos. Si 'Entidad 3D' no encuentra estos nombres exactos, el programa se interrumpirá de forma brusca.
Estos nombres y su función han de ser los siguientes:

- **'Idle'** para una posición estática del personaje.
- **'Shoot'** el personaje disparando.
- **'Walk'** el personaje andando.
- **'Run'** el personaje corriendo.
- **'Die'** el personaje muriendo.
- **'Die_de_frente'** el personaje muriendo de otra forma.
- **'Die_rodilla'** el personaje muriendo de otra forma.
- **'Die_vueltas'** el personaje muriendo de otra forma.
- **'Hit'** el personaje recibe un impacto.

Estos nueve nombres de animaciones sólo son obligatorios para personajes protagonistas y para enemigos.
Para el caso de los personajes secundarios sólo es obligatoria la animación **'Idle'**.
Para los objetos no es necesario ni siquiera que tenga alguna animación.

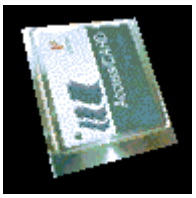
4º- Objetos:

Vamos a continuar este tutorial volviendo a retomar nuestro editor de niveles 'World Editor'.
Seguro que con el tutorial de 'World editor' habrás creado un nivel con todos los ejemplos de ese tutorial, y lo más seguro es que te haya quedado una especie de escenario muy raro y pequeño, pero aún te resultará muy útil en este tutorial de 'Entidad 3D' para seguir experimentando toda clase de cosas antes de pasarlas a un escenario más grande que podría ser el del juego que quieras crear.
Vamos entonces a darle un poco más de vida a ese escenario.

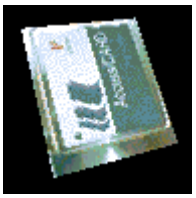
Colocando objetos recogibles:

En 'Entidad 3D' podremos colocar en nuestro escenario una serie de objetos que podrán ser recogidos por el jugador para engrosar un inventario de objetos que nosotros (como creadores del juego) podremos decidir su utilidad en el transcurso del juego.
Estos objetos no tienen el mismo tratamiento que los que representan las armas, munición, energía y escudo, sino que son otros objetos independientes y de utilidad configurable. Cada objeto es en realidad un fichero .act igual que los que hemos utilizado para personajes o también para armas, munición, etc..., pero estos ficheros .act en lugar de estar en la carpeta 'Actors' tienen que estar en la carpeta 'Objects' y tienen que tener un nombre particular para su posterior control.
Para cada objeto distinto debemos tener un .act preparado que será el objeto en 3D que represente ese objeto, en este caso no es necesario que el .act contenga animaciones. También debemos de ir pensando en asociarle un número a cada objeto distinto. Finalmente tenemos que tener preparada una imagen 2D de un tamaño de 32x32 que representará el objeto en el inventario del jugador.
Como podrás ver dentro de la carpeta 'Objects', ya hay unos cuantos objetos que utiliza el juego de ejemplo, si quieres puedes copiarlos con otros nombres para utilizarlos en tus juegos, pero si cambias directamente los nombres de los ficheros, el juego de ejemplo dejará de funcionar. Si haces esto al menos deja el fichero '1_caja.act' ya que siempre lo necesita 'Entidad 3D'. Si borras '1_caja.act', tu juego ni siquiera comenzará.
Vamos a ver como definir uno de estos objetos y donde colocamos la imagen 2D del mismo:
Supongamos que queremos añadir el objeto 'Tarjeta' a nuestra aventura. Ya tenemos un fichero .act que es una tarjeta y también hemos dibujado una imagen de 32x32 de nuestra tarjeta:

Imagen de 32x32



tarjeta.act



Ahora vamos a darle un número a este objeto, por ejemplo el 2, ya que el número 1 ya está ocupado por la caja (por eso se llama '1_caja.act') y que no podemos borrar. Renombramos el archivo .act de la tarjeta al nombre '2_tarjeta.act' y lo copiamos en la carpeta 'Objects' de 'Entidad 3D'.

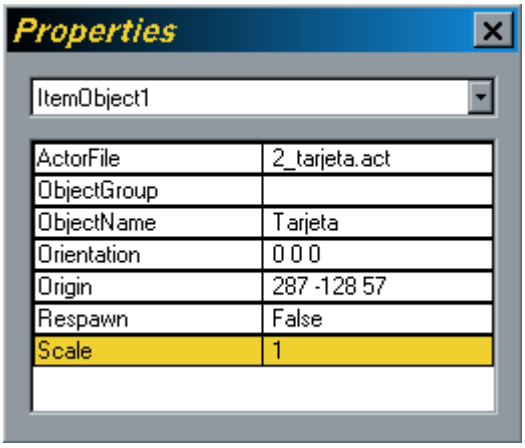
Fichero '2_tarjeta.act' en la carpeta 'Objects'

Seguidamente utilizamos nuestro editor de imágenes preferido (tipo Gimp, Adobe Photoshop o parecido) y abrimos el fichero 'objetos_distintos.bmp' que encontraremos en la carpeta 'Bmp' de 'Entidad 3D'.
Lo que contiene este fichero son todas las representaciones 2D de un tamaño de 32x32 de todos los objetos distintos que tengamos. A medida que los vayamos diseñando los tenemos que ir 'pegando' en esta imagen BMP de arriba a abajo, pegados a la izquierda y sin separación entre ellos. Por lo tanto el primer cuadro de 32x32 en la parte superior izquierda pertenecerá al objeto número 1, que es precisamente la caja (1_caja.act) y para el objeto número 2, que es el objeto que estamos preparando, pondríamos nuestra imagen de 32x32 justo debajo.

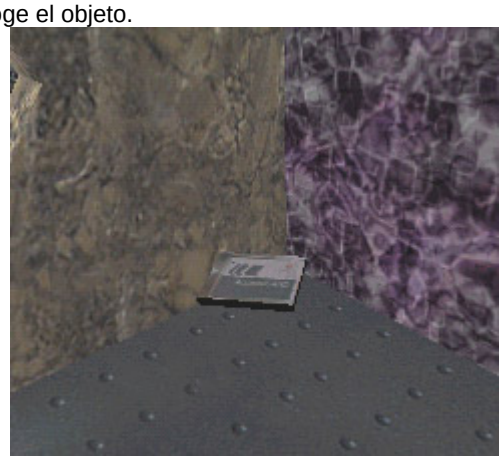
Fichero 'objetos_distintos.bmp' en la carpeta 'Bmp'

Si queremos que este fichero de objetos sea particular de un juego, tendremos que crear un fichero igual que el anterior pero con el nombre ***PrimerNivel*_objetos_distintos.bmp**, siendo 'PrimerNivel' en nombre del primer nivel de nuestro juego. Si hacemos esto siempre tendremos que trabajar sobre este fichero para que los imágenes de los objetos aparezcan en ese juego en concreto.
De esta forma, mediante el nombre del fichero .act y el orden de imágenes en el fichero 'objetos_distintos.bmp', 'Entidad 3D' es capaz de relacionar un objeto .act con su imagen para el inventario, de manera que nosotros no tengamos que hacer nada más.

Ahora vamos a ver como situamos, en nuestro escenario, uno de estos objetos que hemos definido para que el jugador pueda recogerlo con sólo acercarse a él:
Vamos a nuestro escenario en 'World Editor' y añadimos una entidad 'ItemObject' en la posición que queremos que aparezca nuestra tarjeta. Una vez colocada, edita las propiedades de esta entidad y rellena el parámetro 'ActorFile' con '2_tarjeta.act' (sin las comillas). También puedes rellenar el parámetro 'ObjectName' con un nombre para este objeto, que podría ser 'Tarjeta' o 'La llave' o lo que quieras, este nombre sólo lo utiliza 'Entidad 3D' de forma automática para ponerlo en pantalla cuando estemos recogiendo el objeto.



Sólo nos queda compilar el nivel, ver si nuestra tarjeta aparece y probar a ver si nuestro personaje recoge el objeto.

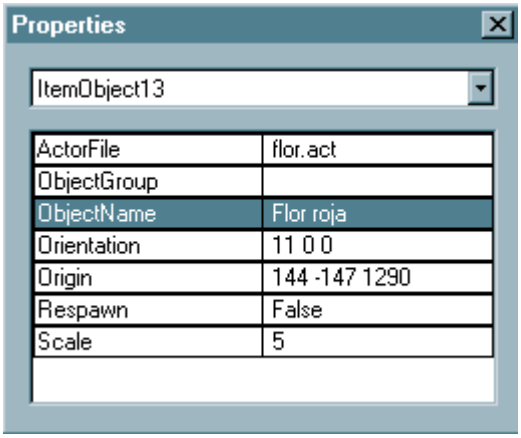


La tarjeta en el escenario

Una vez recogida, la tarjeta aparece en el inventario del jugador

Si queremos podemos añadir en nuestro escenario múltiples objetos iguales, procediendo de la misma forma: añadir entidad 'ItemObject' con 'ActorFile'= 2_tarjeta.act (en el caso de nuestra tarjeta). Al añadir un segundo objeto igual, no es necesario volver a rellenar el parámetro 'ObjectName'.
Pues supongo que ya te podrás imaginar como definir un tercer objeto para nuestro sistema .....

Colocando objetos no recogibles:



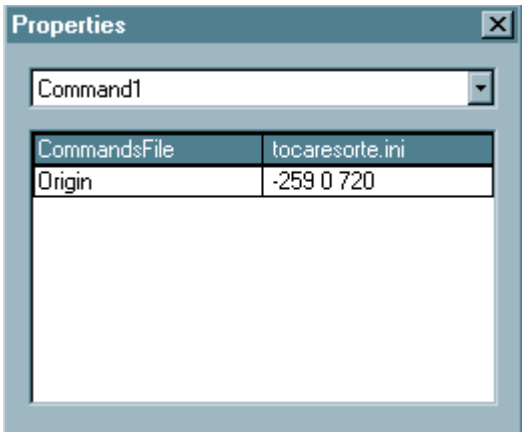
En ocasiones nos puede interesar colocar un objeto representado también por un fichero .act, pero que simplemente esté ahí y que no sea un objeto que el jugador lo pueda recoger.
Pues esto es tan simple como añadir una entidad 'ItemObject' con su 'ActorFile' con el nombre del fichero .act que lo representa, sin números delante ni nada, el nombre tal cual. Este fichero .act también ha de estar en la carpeta 'Objects'.

» Consulta la sección de la entidad 'ItemObject' para conocer a fondo todos sus parámetros.
» Para hacer aparecer/desaparecer el inventario de objetos en pantalla, el jugador debe usar la tecla 'i'.

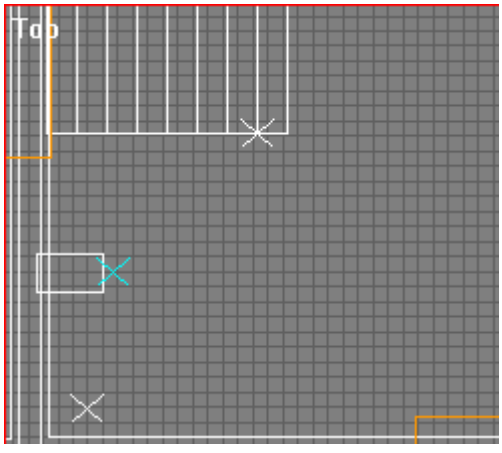
5º- Abriendo puertas:

Tocando resortes:

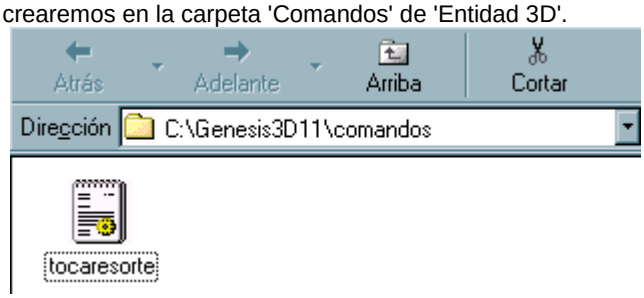
Sin abandonar el editor 'World Editor' vamos a ver como podemos hacer para que una de las puertas que hayamos creado (porque se supone que ya sabemos crear puertas) sólo se abra en ciertas condiciones y no como hasta ahora hemos visto en el tutorial de 'World Editor' que se abrían siempre que alguien se acercaba a ellas, incluido los enemigos.
Para comenzar con algo sencillo vamos a hacer que la puerta se abra siempre y cuando el personaje haya tocado cierto punto del escenario previamente, punto que no tenemos porque poner al lado de la puerta, sino que puede estar en la otra punta del escenario, de la habitación o donde quieras.
El objetivo no es que se abra la puerta al tocar este punto, sino que una vez tocado este punto, cuando el jugador se acerque a la puerta, esta se abrirá siempre.
Para comenzar podemos crear alguna especie de resorte que representará nuestro punto, si quieres puedes hacer un pequeño cubo que sobresalga de una pared sin animación y sin más complicaciones.



Ahora vamos a crear nuestro primer sistema de comandos de 'Entidad 3D' para controlar si el jugador ha tocado o no el resorte y en función de eso, que la puerta se abra o no. Para ello añadimos la entidad 'Command' y la colocaremos en la posición donde hemos colocado el resorte. Si editas las propiedades de esta entidad 'Command' (seguramente se llamará 'Command1') verás un parámetro llamado 'CommandsFile' que rellenaremos con un nombre cualquiera, pero que tenga algún sentido con lo que estamos haciendo, por ejemplo 'tocaresorte.ini' (sin las comillas). Este nombre hará referencia a un fichero llamado 'tocaresorte.ini' que construiremos nosotros mismos con algún editor de textos ANSI tipo Notepad o similar.



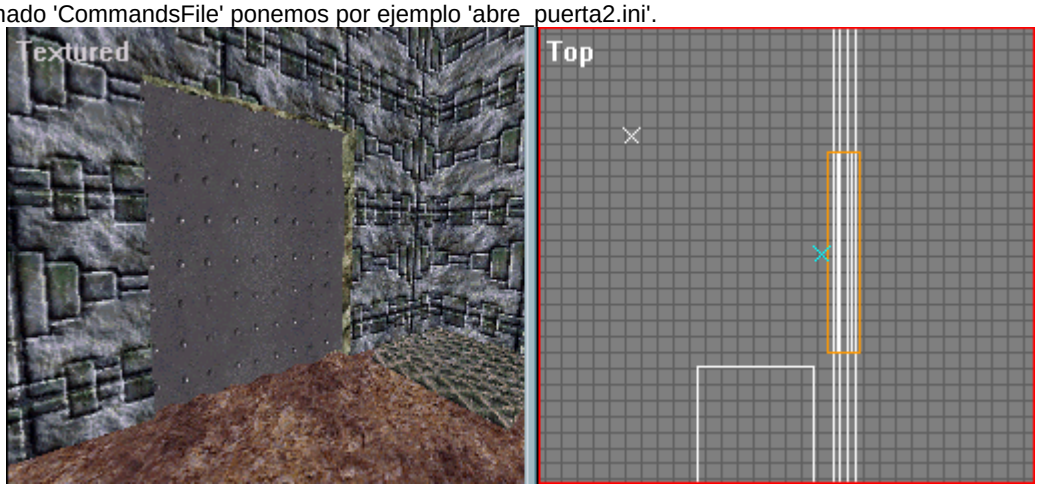
Este fichero con extensión .ini lo crearemos en la carpeta 'Comandos' de 'Entidad 3D'.

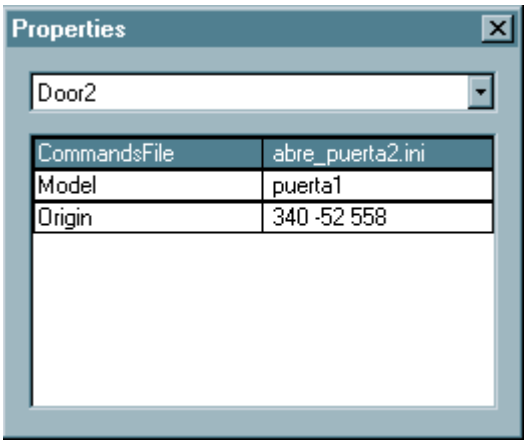


¿ Que hay que poner dentro de este fichero ?... pues todos los comandos y condiciones que queramos que se procesen cuando el jugador alcance la posición donde hemos colocado la entidad 'Command1'. Existen numerosos comandos que se describen con detalle más adelante, pero de momento vamos a centrarnos sólo en los que nos interesan para lo que queremos realizar.
Pues bien, editamos nuestro aún vacío fichero 'tocaresorte.ini' y lo rellenamos con lo siguiente:

```
ACCI PONE tocoresorte=1
```

... y salvamos el fichero. Así de simple. En esta ocasión estamos utilizando el comando 'PONE' para establecer la variable 'tocoresorte' (nombre que nos hemos inventado) a 1 ya que en 'Entidad 3D' cualquier variable está 0 al comenzar el juego. El 'ACCI' siempre hay que ponerlo antes de un comando.
De esta forma cuando el jugador alcance la posición de nuestra entidad 'Command1' se procesará el fichero de comandos 'tocaresorte.ini' con lo que llevará siempre a ponerse a 1 la variable 'tocoresorte'.
¿Y ahora cómo hacemos para que la puerta se abra si el jugador se acerca a ella pero siempre y cuando la variable 'tocoresorte' valga 1?. Pues no nos queda más remedio que hacer otro fichero .ini y asociarlo a la entidad 'Door' que tengamos para esta puerta en cuestión. Editamos esta entidad, que en mi ejemplo se llama 'Door2', y en la casilla del parámetro llamado 'CommandsFile' ponemos por ejemplo 'abre_puerta2.ini'.



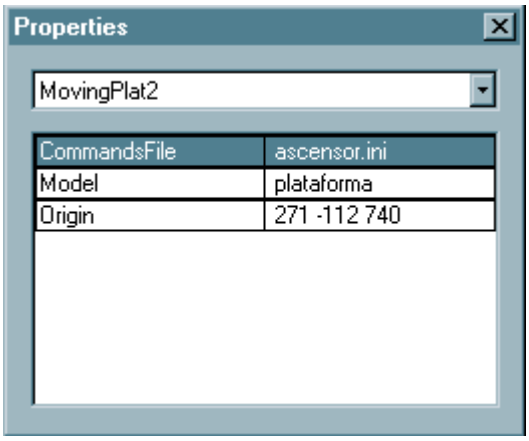


Ahora sólo nos queda crear un fichero llamado 'abre_puerta2.ini' en la carpeta 'Comandos' de 'Entidad 3D'. Dentro de este fichero pondremos lo siguiente:

```
IF COND tocoresorte=1
ACCI TEXT Abriendo puerta
ELSE
ACCI TEXT No has tocado el resorte
ENDIF
```

Como ves he comenzado estableciendo una condición con 'IF COND tocoresorte=1', por lo que todo lo que vaya detrás de esta linea y hasta el indicativo 'ENDIF' afecta a esta condición. En este caso también he utilizado el indicativo 'ELSE' para presentar un mensaje en el caso de que el jugador no haya tocado el resorte.
O sea, que si se cumple la condición se procesarán los comandos entre el 'IF COND' y el 'ELSE', mientras que si no se cumple la condición se procesarán los comandos entre 'ELSE' y 'ENDIF'.
Los comandos 'TEXT' que he puesto son para poner un mensaje en la parte superior de la pantalla. No te olvides nunca de poner un 'ACCI' delante de cada comando.
En nuestro ejemplo la condición se cumplirá siempre y cuando la variable 'tocoresorte' valga 1, además al ser la primera condición de esta fichero .ini, 'Entidad 3D' también lo tomará para decidir abrir la puerta o no.
En resumen, cuando el jugador se acerque a la puerta y la variable 'tocoresorte' valga 1, se abrirá la puerta al mismo tiempo que se verá el mensaje 'Abriendo puerta'. Pero cuando el jugador se acerque a la puerta y la variable 'tocoresorte' no valga 1, pues se presentará el mensaje 'No has tocado el resorte' y la puerta no se abrirá.

Quizás te asalte una cuestión: ¿ Como afectan estos sistemas de comandos a los enemigos ?, ¿ Podría un enemigo llegar a tocar el resorte y luego dirigirse a la puerta y abrirla ?. Pues no, el caso es que los comandos de 'Entidad 3D' no tienen efecto si un enemigo alcanza la posición de una entidad con comandos asociados.



Ya, pero y si el jugador toca el resorte (por lo tanto la variable 'tocoresorte' ya vale 1) y después un enemigo se acerca a la puerta... ¿ la abriría ? ... pues tampoco. La única forma que tendría un enemigo de abrir una puerta es que esa puerta no tenga ningún fichero de comandos .ini asociado.
El funcionamiento de los ascensores o plataformas móviles (entidades 'MovigPlat') es exactamente igual que el de las puertas, sólo que toman el resultado de la primera condición del fichero .ini asociado para mover la plataforma o no.

Utilizando llaves:

Ahora vamos a ver como podemos hacer para que una puerta sólo se abra si el jugador posee la llave correcta.
Para ello vamos a utilizar el objeto tarjeta, que hemos definido en el anterior capítulo de este tutorial, como llave de acceso a una puerta.
También vamos a utilizar la misma puerta que hemos utilizado con el ejemplo del resorte, ya que lo que vamos a hacer es simplemente cambiar los comandos que tenemos en el fichero 'abre_puerta2.ini' que como recordarás esta asociado a la entidad 'Door' de esta puerta.
Abrimos el fichero 'abre_puerta2.ini' con un editor, borramos todo lo que hay e introducimos lo siguiente:

```
IF COND 2_tarjeta.act=1
ACCI TEXT Abriendo puerta
ELSE
ACCI TEXT No tienes la tarjeta
ENDIF
```

Con la condición '2_tarjeta.act=1' lo que estamos preguntando es si el jugador lleva en su inventario de objetos un objeto llamado '2_tarjeta.act', si lleva uno, pues la condición se cumple, si no, pues no se cumple.
Pero piensa una cosa, si el jugador lleva más de una de estas tarjetas, la condición no se cumplirá y la puerta no se abrirá, para solucionar esto tendrías que cambiar la linea del 'IF' anterior por esta:

```
IF COND 2_tarjeta.act>0
```

Diversas variaciones:

A continuación vamos a ver diferentes variaciones de lo que hemos visto en este capítulo para que te hagas una idea de la cantidad de combinaciones que puedes realizar con 'Entidad 3D' si utilizas de forma correcta e ingeniosa diversos comandos y condiciones.

La puerta sólo se abre con la llave pero si previamente se ha tocado el resorte:

```
IF COND tocoresorte=1 COND 2_tarjeta.act=1
ACCI TEXT Abriendo puerta
ELSE
ACCI TEXT Acceso denegado
ENDIF
```

Igual a lo anterior pero además quiero hacer sonar un fichero de audio .MP3 cuando no se abra la puerta:

```
IF COND tocoresorte=1 COND 2_tarjeta.act=1
ACCI TEXT Abriendo puerta
ELSE
ACCI TEXT Acceso denegado
ACCI SONI alarma
ENDIF
```

Para este ejemplo has de tener en la carpeta 'Wav' de 'Entidad 3D' un fichero de sonido llamado 'alarma.mp3'.
De hecho si tienes un fichero llamado 'alarma.wav' también funcionará, o si es 'alarma.mod' también, ya que si te has fijado el comando 'SONI' no necesita la extensión del fichero de sonido, 'Entidad 3D' busca cualquier fichero de sonido 'alarma.' en la carpeta 'Wav' y lo hace sonar.

La puerta sólo se abre con una llave de un solo uso, cuando se abra la puerta, la llave ha de desaparecer del inventario del jugador:

```
IF COND 5_llave.act=1
ACCI TEXT Llave correcta
ACCI QINV 5_llave.act
ELSE
ACCI TEXT No llevas la llave
ENDIF
```

Si el jugador lleva por ejemplo 5 llaves, sólo le desaparecerá una cada vez que abra la puerta.

Un ascensor sólo se mueve si el jugador lleva tres bolas de energía, bolas que han de desaparecer del inventario del jugador al utilizar el ascensor:

```
IF COND 4_bola.act=3
ACCI TEXT Subiendo planta
ACCI QTIN 4_bola.act
ELSE
ACCI TEXT No tienes las 3 bolas
ENDIF
```

Si se abre la puerta, al jugador le desaparecen todas las bolas que pueda llevar en el inventario.

El tema del ascensor y las bolas quiero complicarlo más diciendo al jugador la cantidad de bolas que le faltan en el caso de que no tenga las tres bolas:

```
IF COND 4_bola.act=3
ACCI TEXT Subiendo planta
ACCI QTIN 4_bola.act
ELSE
IF COND 4_bola.act=0
ACCI TEXT Te faltan 3 bolas
ENDIF
IF COND 4_bola.act=1
ACCI TEXT Te faltan 2 bolas
ENDIF
IF COND 4_bola.act=2
ACCI TEXT Solo te falta una bola
ENDIF
ENDIF
```

Quiero realizar dos condiciones distintas, además de controlar lo de la puerta con la tarjeta, quiero advertir al jugador de la necesidad de un objeto en la habitación que quiere entrar:

```
IF COND 2_tarjeta.act=1
ACCI TEXT Abriendo puerta
ELSE
ACCI TEXT No tienes la tarjeta roja
ENDIF

IF COND 7_cristal.act=0
ACCI TEXT Sin el cristal no podrás sobrevivir
ENDIF
```

Estas dos condiciones 'IF COND', y las que quieras, pueden ir en el mismo fichero .ini sin problemas.

6º- Personajes:

Establecer el personaje protagonista:

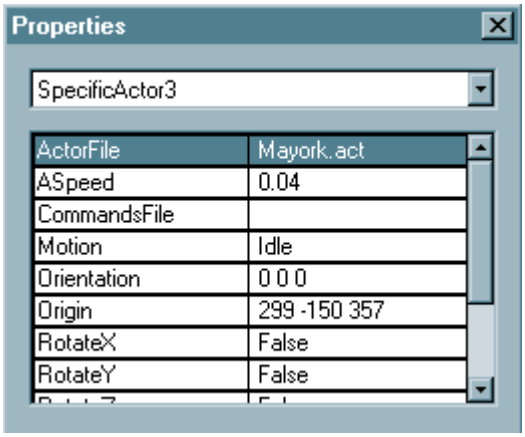
Se trata del personaje que controla el jugador a lo largo del juego. Para establecer que fichero .act es el que representará al protagonista, sólo es necesario copiar ese fichero .act en la carpeta 'Actors' de 'Entidad 3D' pero cambiándole el nombre a: *PrimerNivel*_player1.act.

'PrimerNivel' será el nombre del primer nivel de que consta tu juego.

Para el personaje alternativo el nombre del fichero .act tiene que ser:

*PrimerNivel*_player2.act.

Estos ficheros de personajes serán los que se utilizarán durante todo el juego, independientemente del nivel que esté cargado.



Personajes secundarios:

Son personajes que podremos añadir a nuestro escenario y cuya función tendremos que definir nosotros según el desarrollo de nuestro juego. Estos personajes no se desplazan por el escenario, aunque en el sitio que estén pueden realizar cualquier animación.

Cada uno de estos personajes también lo define un archivo .act que ha de estar en la carpeta 'Actors' de 'Entidad 3D'.

Vamos a ver como podemos poner uno de estos personajes en nuestro escenario.

Nos dirigimos a nuestro programa 'World Editor' y añadimos a nuestro escenario una entidad llamada 'SpecificActor' posicionándola en el lugar que queremos que aparezca el personaje. Ten en cuenta que la cruz que muevas de esta entidad representa los pies del personaje. En la casilla del parámetro 'ActorFile' ponemos en nombre del fichero .act del personaje que deseamos poner, por ejemplo 'Mayork.act'.

Compilamos nuestro escenario y comprobamos el resultado.

La entidad 'SpecificActor' podemos utilizarla para añadir personajes que le digan algo al jugador (por texto o mediante un fichero de sonido), que le entreguen o le quiten algo al jugador, que estén bloqueando una salida o simplemente para dar ambiente al lugar.

Consulta la sección de la entidad 'SpecificActor' para conocer a fondo todos sus parámetros.

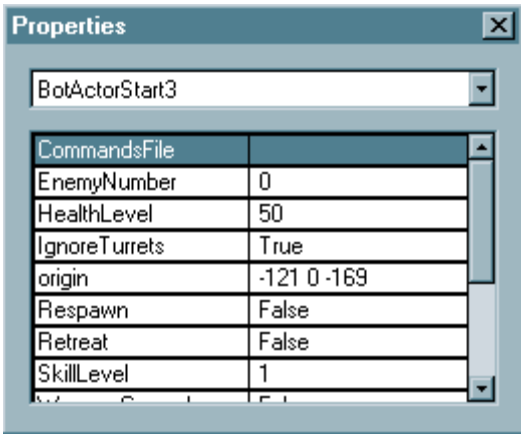
7º- Enemigos:

Añadiendo un enemigo:

En 'Entidad 3D' podemos añadir una serie de personajes (definidos por ficheros .act) que el sistema se encarga de controlar automáticamente y que harán la vida imposible al jugador, al persegirle y dispararle a la mínima oportunidad que tengan al jugador a tiro.
Para añadir uno de estos enemigos lo único que tenemos que hacer es añadir una entidad 'BotActorStart' en la posición que queramos que aparezca este enemigo. Ten en cuenta que la cruz que muevas de esta entidad representa los pies de este enemigo.

Compilamos y comprobamos si nuestro enemigo se lanza hacia nosotros al vernos.

Aquí, como ves, no hemos especificado el nombre del fichero .act que representa el enemigo. Consulta la sección de la entidad 'BotActorStart' para conocer a fondo todos sus parámetros y ver como se establecen los .act que definen a los enemigos.



Entidades disponibles:

DeathMatchStart

Esta entidad se utiliza para posicionar al jugador en el escenario, indicando de esta forma la posición inicial del personaje protagonista al dar comienzo el nivel.
Esta entidad es completamente obligatoria en cualquier nivel, de lo contrario el nivel no dará comienzo.

Parámetros de esta entidad:

Anim_salto
Nombre de la animación para el salto de nuestro personaje protagonista.

Aquí debes introducir el nombre de animación que contenga el fichero .ACT que utilizas para representar el personaje protagonista y que represente el personaje saltando.
Si estás utilizando un personaje que no dispone de esta animación, deja 'Anim_salto' en blanco.
Si utilizas alguno de los personajes descargados de la 'web' de Entidad 3D, puedes rellenar 'Anim_salto' con **JumpLand** ya que este es el nombre de la animación de salto para todos los personajes.

Anim_agachado_Walk

Nombre de la animación de nuestro personaje protagonista avanzando agachado.
Aquí debes introducir el nombre de animación que contenga el fichero .ACT que utilizas para representar el personaje protagonista y que represente el personaje avanzando agachado.
Si estás utilizando un personaje que no dispone de esta animación, deja 'Anim_agachado_Walk' en blanco.
Si utilizas alguno de los personajes descargados de la 'web' de Entidad 3D, puedes rellenar 'Anim_agachado_Walk' con **Walk_agachado** ya que este es el nombre de la animación de esta acción para todos los personajes.

Anim_agachado_Idle
Nombre de la animación de nuestro personaje protagonista agachado y parado.

Aquí debes introducir el nombre de animación que contenga el fichero .ACT que utilizas para representar el personaje protagonista y que represente el personaje agachado y parado.
Si estás utilizando un personaje que no dispone de esta animación, deja 'Anim_agachado_Idle' en blanco.
Si utilizas alguno de los personajes descargados de la 'web' de Entidad 3D, puedes rellenar 'Anim_agachado_Idle' con **Idle_agachado** ya que este es el nombre de la animación de esta acción para todos los personajes.

Anim_escalar
Nombre de la animación de nuestro personaje protagonista escalando.

Aquí debes introducir el nombre de animación que contenga el fichero .ACT que utilizas para representar el personaje protagonista y que represente el personaje escalando.
Si estás utilizando un personaje que no dispone de esta animación, deja 'Anim_escalar' en blanco.

Esta animación se usará automáticamente cuando el jugador escale bloques marcados con 'ladder' en World Editor.

Vidas

Número de vidas que dispone el jugador al dar comienzo el juego.

Jugador_masculino

Se utiliza para personalizar diversas voces que utiliza Entidad 3D de forma automática para el personaje protagonista.
Con valor 'True' se utilizarán voces masculinas.
Con valor 'False' se utilizarán voces femeninas.
Estas voces corresponden a los sonidos de: salto, muerte, daño y respiración al coger salud o escudo.

Salto_jugador

Define la altura a la que saltará el jugador cuando pulse la tecla de salto. Podemos variar este valor entre 0 y 5000, siendo 510 lo normal. Si ponemos 0 el jugador no podrá saltar.

Brillo_jugador

Con un valor que puede variar entre 0 y 255 podemos dar más o menos brillo al personaje protagonista que controlará el jugador en nuestro juego.
En ocasiones los personajes protagonistas son en exceso demasiado oscuros para el nivel que estamos diseñando, por sus texturas o porque el personaje ya es así. Con 'Brillo_jugador' podremos elevar el brillo de las texturas del personaje con sólo variar este parámetro.
Hay que tener en cuenta que este nivel de brillo se mantendrá para todos los niveles que compongan el juego.

Ejemplos del mismo personaje y escenario pero con brillos 120 y 200 para el personaje:

Movimiento_lateral
Con valor 'False' el jugador no podrá desplazarse lateralmente. Se suele utilizar para que cuando el jugador controle por ejemplo un automóvil o moto no pueda realizar desplazamientos laterales que quedarían poco realistas.
Con valor 'True' se permite al jugador moverse lateralmente.
Consulta los comandos LAT0 y LAT1 para cambiar esta característica durante el transcurso del juego.

Avance_direccion
Este parámetro afectará a la forma de controlar al personaje protagonista del juego.
Con valor '**True**' el jugador solo podrá avanzar con las teclas que haya definido para 'adelante', 'atrás', 'derecha' e 'izquierda' (normalmente las flechas del cursor), por lo que el personaje siempre avanzará hacia una de las cuatro direcciones con sólo pulsar una tecla y sin utilizar el ratón.
Con valor '**False**' el jugador avanza con la tecla definida para 'adelante', retrocede con la tecla 'atrás', va de lado hacia la derecha con la tecla 'derecha' y va de lado hacia la izquierda con la tecla 'izquierda'. El jugador avanzará hacia la dirección a la que esté mirando en ese momento. Para variar hacia donde mira el personaje se debe utilizar el movimiento del ratón.
Con valor 'True' sólo funciona este tipo de movimiento con las cámaras tipo Plataformas y con las cámaras fijas, el resto de cámaras, por ser cámaras que rotan para seguir al personaje, este tipo de movimiento es completamente confuso y Entidad 3D ya no lo permite de forma automática.
El tipo de movimiento que se establece con valor 'False' funciona con todos los tipos de cámaras.

Colisión_player1n

Permite ampliar la zona de colisión del personaje protagonista con las paredes. En este parámetro tendremos que entrar el tamaño que deseamos ampliar en sentido negativo para los ejes X Y Z. Los ejes cuyo valor queden a 0 no sufrirán variaciones.

Colisión_player1p
Igual que el parámetro 'Colisión_player1n' pero en sentido positivo.

Colisión_player2n
Igual que 'Colisión_player1n' pero para el personaje protagonista 2 (el que se activa con 'ACCI PERS 2').

Colisión_player2p
Igual que 'Colisión_player1p' pero para el personaje protagonista 2 .
Normalmente, si utilizas personajes normales no tendrás que realizar ninguna variación de la zona de colisión del personaje, pero si utilizas 'personajes' especiales o simplemente porque deseas hacer algo especial, tendrás que ir probando variaciones hasta que obtengas el resultado deseado.
A continuación puedes ver un ejemplo de un vehículo que se hundía en las paredes al chocar con ellas, pero después de aplicar valores de 200 en X y Z positivos y negativos, el vehículo ya chocaba bien:

Sonido_pasos
Con valor 'True' se oirán los pasos del jugador al correr.
Con valor 'False' no se oirán los pasos del jugador al correr.

Pierde_armas
Con valor 'True' el jugador perderá todas sus armas y municiones cada vez que muera.
Con valor 'False' el jugador no perderá todas sus armas y municiones cada vez que muera.

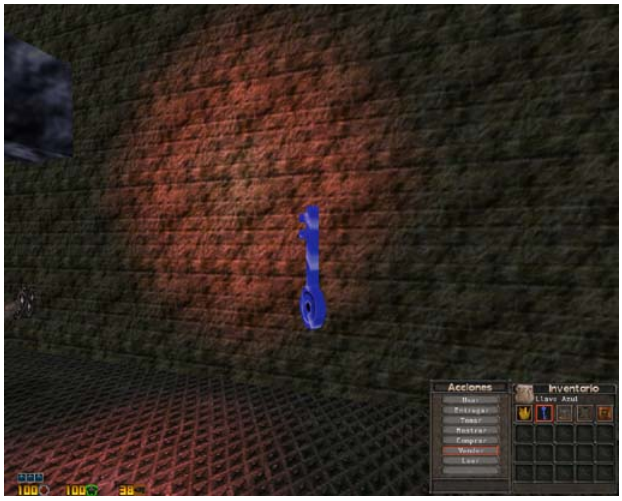
Usar_Blaster
Con valor 'True' el jugador podrá utilizar desde el primer momento el arma número 1 (Blaster).
Con valor 'False' el jugador no podrá utilizar el arma Blaster en todo el juego.

Ver_arma_1
Con valor 'True' y la entidad VerArma en el mapa, el jugador podrá ver, en la vista en primera persona, el arma que está utilizando.
Con valor 'False' el jugador no verá el arma que está utilizando.

Sangre
Con valor 'True' el jugador presentará una animación de sangre al recibir impactos de las armas número 1 y 4.
Con valor 'False' los impactos sólo se representarán como pequeñas explosiones.

Sangre_charco
Con 'True' aparecerá un charco de sangre debajo del jugador cuando esté muerto.
Con 'False' no aparecerá ningún charco de sangre debajo del jugador cuando esté muerto.

Inventario_3D
Con 'True' el objeto que seleccione el jugador de su inventario de objetos, aparecerá enfrente del jugador en 3D y rotando, en la vista en primera persona:



En el caso de vistas en tercera persona, el objeto aparecerá encima del jugador:

Esto sólo funciona en objetos recogidos por el jugador como entidades 'ItemObject' y que no reaparezcan, o bien un objeto entregado al jugador con el comando 'PINV' pero que ya tenga alguno igual en su inventario que haya sido recogido como objeto 'ItemObject'.
Con 'False' en este parámetro los objetos del inventario no se mostrarán en 3D en ningún momento.

Dinero
Cantidad de dinero que tendrá el jugador al dar comienzo el nivel.

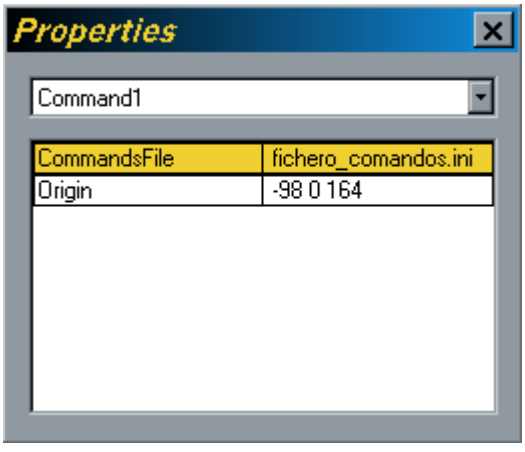
Velocidad
Define la velocidad a la que podrá desplazarse el jugador. Podemos variar este valor entre 0 y 10000.

QuitaSaludCaidas
Resta salud al jugador en las caídas. La cantidad de salud que resta depende de la altura desde donde haya caído.
Con 'True' resta salud.
Con 'False' no resta salud.

Sombra
De 0 a 2.
Establece el tipo de sombra que tendrá el jugador en todo el juego.
Valor **0** sin sombra.
Valor **1** con sombra en círculo.
Valor **2** con sombra dinámica.
Si quieres ver ejemplos de estos tipos de sombras, consulta la sección de la entidad ConfigEnemigos.

Consulta la sección 'Nuestra primera habitación' del tutorial de 'World Editor' para saber como colocar correctamente esta entidad.

Command



Esta entidad se utiliza para provocar el proceso de comandos al pasar el jugador por un punto en concreto del escenario.

Parámetros de esta entidad:

CommandsFile
Introduciremos el nombre del fichero de comandos .ini que contiene los comandos que deseamos procesar cuando el jugador pase por la posición donde hemos situado esta entidad. Este fichero .ini ha de estar en la carpeta 'Comandos' de 'Entidad 3D'.

Origin
Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Boton_accion

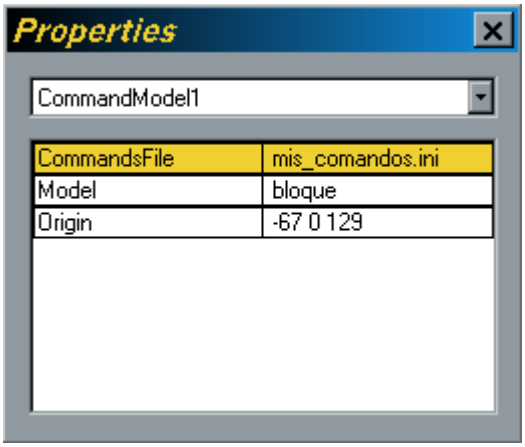
Con valor 'True', los comandos contenidos en el fichero especificado en 'CommandsFile' se procesarán si el jugador se sitúa en la posición donde hemos situado esta entidad y al mismo tiempo pulsa el botón o tecla 'Acción'.
Con valor 'False', sólo será necesario que el jugador se sitúe en la posición de la entidad para procesar los comandos que contiene el fichero .ini que hemos especificado en 'CommandsFile'.

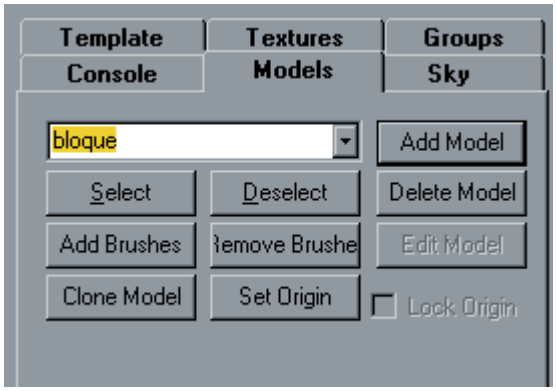
Visible

Con 'True', en la posición que ocupa esta entidad en el nivel, cuando juegues al nivel, verás una bola verde que te podrá servir de referencia para saber la posición exacta de esta entidad mientras desarrollas el juego.
Con 'False', la entidad será invisible.
No te olvides de poner a 'False' este parámetro en todas las entidades 'Command' de todos los niveles en la versión final de tu juego para que no le aparezcan al jugador.

» Consulta la sección 'Abriendo puertas' para ver ejemplos de como usar esta entidad utilizando diversos comandos.

CommandModel





Esta entidad se utiliza para provocar el proceso de comandos cuando el jugador toque un 'model' del escenario. Su funcionamiento es similar al de la entidad 'Command' pero en lugar de establecer un punto en concreto para procesar comandos, con la entidad 'CommandModel' lo hacemos con todo un 'model' (conjunto de 'brushes').

Parámetros de esta entidad:

CommandsFile

Introduciremos el nombre del fichero de comandos .ini que contiene los comandos que deseamos procesar cuando el jugador toque el 'model' que hemos establecido. Este fichero .ini ha de estar en la carpeta 'Comandos' de 'Entidad 3D'.

Model

Selecciona aquí el 'model' que quieres que sea el que ha de tocar el jugador para que se procesen los comandos que contiene el fichero .ini definido en 'CommandsFile'.
Recuerda que estos 'models' son los que tenemos definidos en la siguiente pestaña de 'World Editor':
Es imprescindible que este parámetro no se quede 'null', de lo contrario tu juego dejará de funcionar de forma brusca.

CommandsFile2

Introduciremos el nombre del fichero de comandos .ini que contiene los comandos que deseamos procesar cuando el jugador dispare sobre el 'model' que hemos establecido. Este fichero .ini ha de estar en la carpeta 'Comandos' de 'Entidad 3D'.
Sólo funciona con las armas número 1 (Blaster) y número 3 (Lanza-cohetes) del jugador, no de los enemigos.

Origin

Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

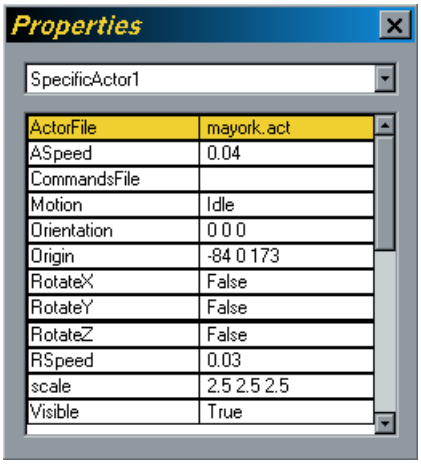
Boton_accion

Con valor 'True', los comandos contenidos en el fichero especificado en 'CommandsFile' se procesarán si el jugador toca el 'model' que se ha establecido y al mismo tiempo pulsa el botón o tecla 'Acción'.
Con valor 'False', sólo será necesario que el jugador toque el 'model' para procesar los comandos que contiene el fichero .ini que hemos especificado en 'CommandsFile'.

Consulta la sección 'Abriendo puertas' para ver ejemplos de como usar esta entidad utilizando diversos comandos.

SpecificActor

Esta entidad se utiliza para posicionar personajes secundarios en el escenario.



Parámetros de esta entidad:

ActorFile

Introducir aquí el nombre del fichero .act que quieras situar como personaje en el escenario. Este fichero .act ha de estar en la carpeta 'Actors' de 'Entidad 3D'.
***Nota**: Si utilizas el editor de niveles 'World Editor Pro', los nombres de los ficheros .act que indiques en este parámetro 'ActorFile', ponlos con las extensión .ACT en mayúsculas (por ejemplo 'estatua.ACT') si quieres que el modelo 3D de este fichero .act no aparezca en el editor y quieras que el propio editor funcione más rápido, ya que si indicas la extensión .act del fichero en minúsculas (por ejemplo 'estatua.act') verás una representación del modelo 3D con una sola textura pero si tienes muchas entidades 'SpecificActor' en tu nivel, observarás que el editor se hace muy lento a la hora de modificar cualquier elemento*.

Motion

Nombre de la animación que queremos que realice este personaje. Este nombre ha de ser uno de los nombres de animación que contiene particularmente cada fichero .act. Utiliza la utilidad 'Actor View' para ver los nombres de animación del .act que te interese.
La animación que seleccionemos se realizará de forma continua mientras este personaje esté en el escenario.

ASpeed

Variando este valor podremos regular la velocidad de la animación que hemos establecido en 'Motion'.

Rotate X

Con 'True' haremos que el personaje realice una rotación continua sobre su eje X.

Rotate Y

Con 'True' haremos que el personaje realice una rotación continua sobre su eje Y.

Rotate Z

Con 'True' haremos que el personaje realice una rotación continua sobre su eje Z.

RSpeed

Variando este valor podremos regular la velocidad de rotación del personaje en el caso que la hayamos establecido con Rotate X, Rotate Y o Rotate Z.

CommandsFile

Introduciremos el nombre del fichero de comandos .ini que contiene los comandos que deseamos procesar cuando el jugador toque al personaje. Este fichero .ini ha de estar en la carpeta 'Comandos' de 'Entidad 3D'.

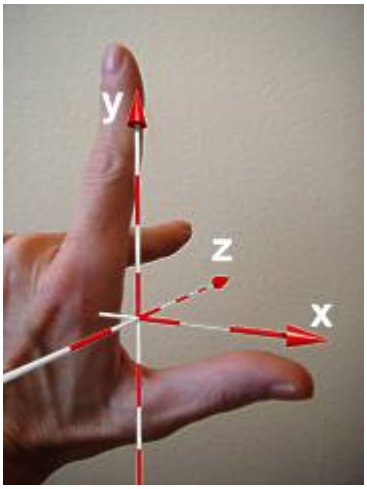
Boton_accion

Con valor 'True', los comandos contenidos en el fichero especificado en 'CommandsFile' se procesarán si el jugador toca al personaje y al mismo tiempo pulsa el botón o tecla 'Acción'.
Con valor 'False', sólo será necesario que el jugador toque al personaje para procesar los comandos que contiene el fichero .ini que hemos especificado en 'CommandsFile'.

Origin

Son tres cifras (X Y Z) separadas por un espacio que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Orientation



Son tres cifras (X Y Z) que determinan la orientación que presentará el objeto en el escenario. Variando estos tres ángulos (con valores entre 0 y 365 grados) podremos hacer aparecer nuestro objeto o personaje boca abajo, tumbado, ladeado, etc..
Si ponemos 0 0 0 el objeto se presentará en la orientación inicial de diseño de este objeto 3D.

Ejes X Y Z en Entidad 3D

Scale

Son tres cifras (escala en X(ancho), escala en Y(alto), escala en Z(largo)) separadas por un espacio. Permiten agrandar o reducir el tamaño inicial de nuestro personaje. Si ponemos 1 aparecerá en el escenario en el tamaño inicial de diseño de este objeto 3D. Si ponemos 2 aparecerá el doble de grande. Si ponemos 0.5 aparecerá a la mitad de su tamaño.

Visible

Con 'False' haremos que este personaje no aparezca en el escenario. Esto se utiliza en casos en que nos pueda interesar definir la posición de un personaje, pero que al dar comienzo el juego aún no aparezca y que lo hagamos aparecer según el desarrollo de nuestro juego. Esto último lo podremos hacer con el comando 'VIS1'.

Bloqueo_ver

Con valor 'True', nos aparecerá un rectángulo azul transparente que envolverá el objeto o personaje. Esta zona azul es la zona de bloqueo para este 'actor' y representa el espacio que el jugador no podrá atravesar mientras el 'actor' sea visible. Tanto si este valor está como 'True' o como 'False', el jugador no podrá atravesar esta zona.
A continuación puedes ver un ejemplo de un tanque con este rectángulo azul activado:

Bloqueo_X Bloqueo_Y Bloqueo_Z
En estas tres cajas podemos especificar valores para variar la zona de bloqueo (visible si pones 'True' en Bloqueo_ver), con el objetivo de modificar el tamaño que tiene esta zona por defecto. Si utilizas valores positivos aumentarás de tamaño la zona, mientras que si utilizar valores negativos reducirás la zona.

Brillo
Con un valor que puede variar entre 0 y 255 podemos dar más o menos brillo al modelo 3D que represente este 'SpecificActor'.

Bloqueo
Con valor 'True', el jugador colisionará con este objeto o personaje.
Con valor 'False', el jugador atravesará este objeto o personaje.
Con 'False' nos deja la libertad de poder controlar las zonas de bloqueo de este objeto mediante bloques que definamos en el escenario (en World Editor) en el mismo punto donde tengamos situado el 'SpecificActor'.
Para que estos bloques realicen la tarea de bloquear al jugador pero que al mismo tiempo no se vean, debemos acceder a las propiedades de todas las caras de ese 'brush' (ventana 'Faces Attributes'), marcar 'Transparent' y finalmente poner el valor 0 en 'Transparency Value'.

Transparencia
De 0 a 255.
Nivel de transparencia del objeto o personaje.
255 opaco, 0 completamente transparente, 128 semi-transparente.

RenderSiNoVisible
Con valor 'False', si el modelo 3D .ACT no está a la vista del jugador, el programa no realizará los cálculos necesarios para visualizar este modelo. Esto permite aumentar ligeramente el rendimiento del programa en algunas ocasiones.
Con valor 'True', los cálculos para visualizar el modelo 3D .ACT se realizarán siempre que el jugador lo tenga en frente, aunque delante del modelo haya una pared que le impida verlo.
Intenta siempre poner este parámetro a 'False' si no causa algún efecto no deseado en tu juego.
Una situación en la que no podrás poner este parámetro a 'False' es por ejemplo cuando pones un 'SpecificActor' en tu nivel y en su interior has situado un bloque sólido en 'World Editor' para que el jugador colisione con él. En ese caso si pones 'False' en 'RenderSiNoVisible' nunca verás el modelo 3D, por lo que debes poner 'True' para verlo siempre.

Sombra
De 0 a 2.
Establece el tipo de sombra para este objeto.
Valor **0** sin sombra.
Valor **1** con sombra en círculo.
Valor **2** con sombra dinámica.
Si quieres ver ejemplos de estos tipos de sombras, consulta la sección de la entidad ConfigEnemigos.

Consulta la sección 'Personajes' para ver un ejemplo de como añadir una entidad de este tipo.

Personajes interactivos:

Si no queremos que nuestros personajes sean simples 'estatuas' que no se relacionan con el jugador, tendremos que asociar la entidad 'SpecificActor' de nuestros personajes con algún fichero de comandos .ini para que realicen ciertas acciones según las circunstancias.
Lo que pueden hacer estos personajes lo determinan los comandos disponibles en 'Entidad 3D', desde decirle algo al jugador (mediante un texto), hasta darle algo al jugador o hacer sonar un fichero de audio con alguna voz grabada.
A continuación, y a modo de ejemplo, vamos a crear un fichero .ini en la carpeta 'Comandos' y que estableceremos como nombre de fichero .ini en 'CommandsFile' para una entidad de este tipo. El fichero en cuestión contendría lo siguiente:

```
IF COND llave_entregada=0
IF COND 5_dinero.act=1
ACCI DIAL vende_llave
ACCI QINV 5_dinero.act
ACCI PINV 4_llave.act
ACCI PONE llave_entregada=1
ELSE
ACCI DIAL dame_dinero
ENDIF
ELSE
ACCI TEXT No tengo nada mas de tu interes
ENDIF
```

¿ Que hace todo esto ?. Lo que pretendemos hacer es que un personaje le venda una llave al jugador si este lleva dinero en su inventario. Evidentemente, para este ejemplo necesitaríamos los objetos '5_dinero.act' (para el dinero) y '4_llave.act' (para

la llave). También vamos a preparar un par de respuestas por parte del personaje, una para cuando le entregue la llave al jugador y otra para cuando el personaje le pida dinero al jugador a cambio de la llave.
Estos diálogos han de estar en forma de ficheros en la carpeta 'Comandos' y con la extensión .dlg. Crearemos el fichero 'vende_llave.dlg' conteniendo las líneas siguientes:

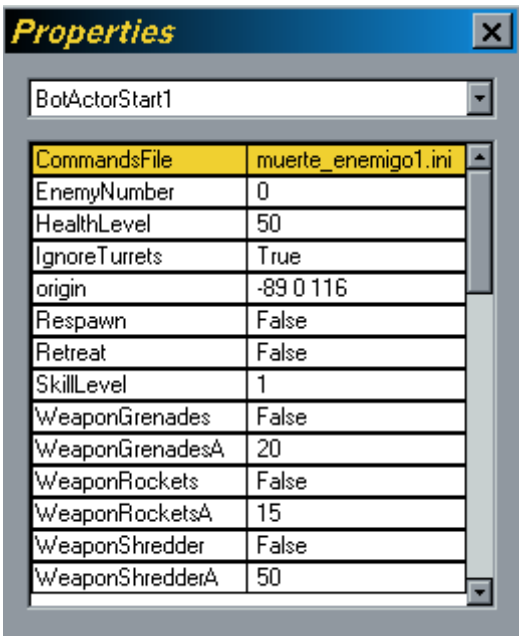
Gracias por tu dinero, te aseguro que lo has
empleado bien. Toma la llave del salon Rojo
y ten cuidado con los guardianes.

También crearemos el fichero 'dame_dinero.dlg' con lo siguiente:

Si estas buscando la llave del salon Rojo te
aseguro que no te saldra gratis. Aqui sin
dinero con conseguiras nada.
Ya estamos en condiciones de entender el grupo de comandos anterior.
Recordarte que estos comandos se procesarán sólo cuando el jugador intente tocar al personaje.
En primer lugar, decirte que utilizo la variable 'llave_entregada' (que me he inventado) para controlar la entrega de la llave al jugador. De esta forma podré presentarle al jugador diversos mensajes en función del valor de esta variable.
El primer 'IF', comprueba la condición 'llave_entregada=0', lo utilizo para, si no se cumple significará que ya le hemos entregado la llave al jugador, pues le presentamos un mensaje rápido a través del comando 'ACCI TEXT' poniéndole 'No tengo nada mas de tu interes'. Este mensaje aparecerá en la parte superior de la pantalla en azul y desaparecerá en pocos segundos. Como ves, para este mensaje no hemos creado un fichero .dlg ya que opino que para un mensaje tan corto no era necesario.
Pero volvamos otra vez a la condición que estamos analizando pero esta vez supongamos que la condición se cumple y la variable 'llave_entregada' vale 0 (que sería lo que valdría al comenzar el juego), pues entonces aún comprobamos otra cosa más, ya que el jugador podría llevar el dinero o no. Si lo lleva la condición 'COND 5_dinero.act=1' se cumplirá y por lo tanto se desencadenarán todos los comandos hasta el siguiente 'ELSE', comandos que hacen lo siguiente:
ACCI DIAL vende_llave
(pone en pantalla el diálogo que hemos preparado en el fichero 'vende_llave.dlg')
ACCI QINV 5_dinero.act
(quita el objeto '5_dinero.act' del inventario del jugador)
ACCI PINV 4_llave.act
(pone el objeto '4_llave.act' en el inventario del jugador)
ACCI PONE llave_entregada=1
(pone la variable 'llave_entregada' a 1)
Finalmente si no se cumple la condición 'COND 5_dinero.act=1' es que el jugador no lleva el dinero y por lo tanto no le damos nada y con el comando 'ACCI DIAL dame_dinero' le presentamos el diálogo que tenemos preparado en el fichero 'dame_dinero.dlg'.
Ya te estarás haciendo una idea de la cantidad de posibilidades que tienes con estos ficheros de comandos .ini, y no sólo en entidades 'SpecificActor' sino en todas las entidades de 'Entidad 3D' que soportan estos ficheros.
Si combinas adecuadamente comandos y condiciones dentro de otras condiciones, etc... podrás obtener gran cantidad de complejas situaciones que podrán adaptarse a las necesidades del guión de tu juego. Al principio te puede parecer un poco liado, pero con la práctica, si ordenadas bien los comandos y los escalonas correctamente para no ver toda una cascada confusa de comandos, verás que ni de lejos tiene la complejidad que te supondría realizar lo mismo programando en lenguajes de programación.

BotActorStart

Esta entidad se utiliza para posicionar enemigos o personajes que se desplazarán por el escenario.
Los .act que representan los enemigos/personajes han de estar con los siguientes nombres en la carpeta 'Actors':



*PrimerNivel*_enemigo1.act
*PrimerNivel*_enemigo2.act
*PrimerNivel*_enemigo3.act
*PrimerNivel*_enemigo4.act
*PrimerNivel*_enemigo5.act
*PrimerNivel*_enemigo6.act
*PrimerNivel*_enemigo7.act
*PrimerNivel*_enemigo8.act

... donde *PrimerNivel* es el nombre del primer nivel de tu juego.
Por ejemplo, si tenemos un juego que consta de 3 niveles (puente.bsp, carretera.bsp y escape.bsp) los ficheros .act de los enemigos tendrían que tener estos nombres:
puente_enemigo1.act
puente_enemigo2.act
puente_enemigo3.act
puente_enemigo4.act
puente_enemigo5.act
puente_enemigo6.act
puente_enemigo7.act
puente_enemigo8.act

Estos 8 enemigos se utilizarán para todos los niveles de que conste el juego.

'Entidad 3D' te permite de esta forma tener hasta 8 tipos de enemigos distintos para todo el juego con sólo dejar los nombres de los fichero .act en ese formato.
Si no tienes o no quieres tantos tipos de enemigos, copia entonces algún enemigo de los que tengas con estos nombres, el caso es tener siempre estos nombres de archivo con algún enemigo, sean distintos o no.
Dentro de tu escenario podrás repetir enemigos y por lo tanto poner cuantos quieras de un mismo tipo.
Si los nombres de los enemigos no comienzan con el nombre del primer nivel del juego o con nombres del nivel, Entidad 3D utilizará 8 ficheros .act provisionales con los nombres (**enemigo1.act, enemigo2.act, enemigo3.act, enemigo4.act, enemigo5.act, enemigo6.act, enemigo7.act y enemigo8.act**) que se encuentran en la carpeta 'Actors'.
Si deseas que algún enemigo, o todos, cambien en un nivel en concreto tendrás que dejar en la carpeta 'Actors' esos enemigos con los siguientes nombres:
*Nivel*_enemigo1.act
*Nivel*_enemigo2.act
*Nivel*_enemigo3.act
*Nivel*_enemigo4.act
*Nivel*_enemigo5.act
*Nivel*_enemigo6.act
*Nivel*_enemigo7.act
*Nivel*_enemigo8.act
... donde *Nivel* es el nombre del nivel donde quieres realizar el cambio.
Por ejemplo, si tenemos un juego que consta de los 3 niveles del ejemplo anterior (puente carretera y escape) los ficheros .act de dos enemigos que quisiéramos cambiar (por ejemplo el 4 y el 6) del nivel 'carretera' tendrían que tener estos nombres:
carretera_enemigo4.act
carretera_enemigo6.act
... en cuyo caso en el nivel 'carretera' los enemigos número 4 y número 6 se sustituirían por los modelos de los ficheros **carretera_enemigo4.act** y **carretera_enemigo6.act** respectivamente.

Parámetros de esta entidad:

CommandsFile
Introduciremos el nombre del fichero de comandos .ini que contiene los comandos que deseamos procesar cuando el enemigo muera. Este fichero .ini ha de estar en la carpeta 'Comandos' de 'Entidad 3D'.
Procesar comandos una vez muere un enemigo en concreto nos puede servir por ejemplo para cambiar un valor de una variable que en algún punto de nuestro escenario comprobamos, por ejemplo, para dejar abrir una puerta o para entregar un objeto.

EnemyNumber
Si ponemos 0 , el enemigo que aparecerá será uno al azar entre los 6 primeros enemigos.
Si ponemos 1 , el enemigo que aparecerá será el del archivo '*PrimerNivel1*_enemigo1.act'.
Si ponemos 2 , el enemigo que aparecerá será el del archivo '*PrimerNivel2*_enemigo2.act'.
Si ponemos 3 , el enemigo que aparecerá será el del archivo '*PrimerNivel3*_enemigo3.act'.
Si ponemos 4 , el enemigo que aparecerá será el del archivo '*PrimerNivel4*_enemigo4.act'.
Si ponemos 5 , el enemigo que aparecerá será el del archivo '*PrimerNivel5*_enemigo5.act'.
Si ponemos 6 , el enemigo que aparecerá será el del archivo '*PrimerNivel6*_enemigo6.act'.
Si ponemos 7 , el enemigo que aparecerá será el del archivo '*PrimerNivel7*_enemigo7.act'.
Si ponemos 8 , el enemigo que aparecerá será el del archivo '*PrimerNivel8*_enemigo8.act'.

HealthLevel
Nivel de salud inicial del enemigo, puede ir desde 1 a 100. A más valor más salud.

Respawn
Con 'True', el enemigo vuelve a reaparecer, tras unos segundos, después de que el jugador lo haya matado.
Con 'False', el enemigo no vuelve a reaparecer una vez el jugador lo haya matado.

WeaponBlaster
Con 'True' el enemigo dispone del arma número 1 desde el primer momento.

WeaponBlasterA
Cantidad de munición, para el arma número 1, de que dispone el enemigo al dar comienzo el juego. Este valor puede ir desde 1 a 255.

WeaponGrenades
Con 'True' el enemigo dispone del lanza-granadas desde el primer momento.

WeaponGrenadesA
Cantidad de munición (granadas en este caso) de que dispone el enemigo al dar comienzo el juego. Este valor puede ir desde 1 a 255.

WeaponRockets
Con 'True', el enemigo dispone del lanza-cohetes desde el primer momento.

WeaponRocketsA
Cantidad de munición (cohetes en este caso) de que dispone el enemigo al dar comienzo el juego. Este valor puede ir desde 1 a 255.

WeaponShredder
Con 'True' el enemigo dispone del fusil desde el primer momento.

WeaponShredderA
Cantidad de munición (balas en este caso) de que dispone el enemigo al dar comienzo el juego. Este valor puede ir desde 1 a 255.

Origin
Son tres cifras (X Y Z) que determinan la posición del enemigo en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

IgnoreTurrets
'sin función por el momento'.

Retreat
'sin función por el momento'.

SkillLevel
Determina la frecuencia de disparo del enemigo. El valor puede valer entre 1 y 10, siendo 1 la frecuencia de disparo más alta (enemigo más agresivo).

Static
Con valor 'True' el enemigo no se moverá de su posición, aunque si que apuntará al jugador y le disparará si lo tiene a tiro.

Se_detiene
Con valor 'True' el enemigo no está corriendo continuamente, sino que de vez en cuando se detiene.

Tiempo_reaparicion
Una vez haya muerto este enemigo, establece el tiempo (en segundos) que ha de transcurrir para que este enemigo vuelva a reaparecer.
El parámetro 'Respawn' ha de estar a 'True' para que funcione este tiempo.

Quita_salud
Cantidad de salud que se restará al jugador al tocar este enemigo.
Con valor 0, este enemigo no resta salud al jugador si lo toca.

Velocidad
Esta cifra regula la velocidad a la que se desplaza este enemigo por un nivel en concreto. A más valor mayor velocidad.
Ten en cuenta que si se trata de un enemigo que corre, la animación de correr será proporcional a la velocidad que hayas establecido.
3700 es el valor normal y que aparece por defecto para la velocidad.

Dispara
Con valor 'True' este enemigo disparará con las armas de las que disponga.
Con valor 'False' este enemigo no disparará en ningún caso, ni siquiera con el arma 1 Blaster.

Inicio_parado
Con valor 'True' este enemigo no se moverá cuando comience el nivel, sólo comenzará a moverse cuando vea al jugador.
Con valor 'False' este enemigo se moverá desde el momento que comience el nivel, independientemente de si ve al jugador o no.

Cobarde
Con valor 'True' este enemigo huirá del jugador cuando lo vea. Tampoco le disparará.
Con valor 'False' no será cobarde y su comportamiento dependerá de lo que hayas establecido en otros modos de comportamiento en esta misma entidad.

CobardePorc
Porcentaje de tiempo en modo cobarde (de 0 a 100). Poner 0 para todo el tiempo.
Este porcentaje sólo se aplica si el parámetro 'Cobarde' está 'True'.
Fuera del porcentaje establecido, su comportamiento dependerá de lo que hayas establecido en otros modos de comportamiento en esta misma entidad.

Indiferente
Con valor 'True' este enemigo dará vueltas no muy lejos del jugador. Tampoco le disparará.
Con valor 'False' no será indiferente y su comportamiento dependerá de lo que hayas establecido en otros modos de comportamiento en esta misma entidad.

IndiferentePorc
Porcentaje de tiempo en modo indiferente (de 0 a 100). Poner 0 para todo el tiempo.
Este porcentaje sólo se aplica si el parámetro 'Indiferente' está 'True'.
Fuera del porcentaje establecido, su comportamiento dependerá de lo que hayas establecido en otros modos de comportamiento en esta misma entidad.

Buscar
Con valor 'True' este enemigo buscará al jugador evitando obstáculos.
Con valor 'False' este enemigo no buscará al jugador, pero eso no quiere decir que no le dispare cuando le vea.
Este funcionamiento del enemigo no es un modo en sí, y puede funcionar simultáneamente con los modos Indiferente o Ataque.

BuscarPorc
Porcentaje de tiempo que este enemigo se dedicará a buscar al jugador (de 0 a 100). Poner 0 para todo el tiempo.
Este porcentaje sólo se aplica si el parámetro 'Buscar' está 'True'.

Tiempo_corriendo
Con el parámetro 'Cobarde' en 'False', tiempo en segundos que el enemigo se moverá por el escenario buscando al jugador cuando este no esté a la vista. Transcurrido ese tiempo el enemigo permanecerá parado esperando a ver al jugador.
Con el parámetro 'Cobarde' en 'True', tiempo en segundos que el enemigo huirá del jugador cuando este no lo tenga a la vista. Transcurrido ese tiempo el enemigo permanecerá parado hasta que vea al jugador.

Abalanzarse
Con valor 'True' este enemigo, durante el ataque, intentará llegar hasta la misma posición que se encuentra en jugador sin mantener ninguna distancia de seguridad entre él y el jugador. Este modo es ideal para enemigos que infringen daño con el contacto como pueden ser zombis o fantasmas.
Con valor 'False' este enemigo, durante el ataque, mantendrá constantemente una distancia mientras ataca al jugador.

Masculino
Se utiliza para personalizar diversas voces que utiliza Entidad 3D de forma automática para los enemigos.
Con valor 'True' se utilizarán voces masculinas.
Con valor 'False' se utilizarán voces femeninas.
Los sonidos que emiten los/las enemigos/as se definen en la entidad 'ConfigSonidosEne'.

Sangre
Con valor 'True' este enemigo presentará una animación de sangre al recibir impactos de las armas número 1 y 4.
Con valor 'False' los impactos sólo se representarán como pequeñas explosiones.

Tiempo_sonido
Intervalo de tiempo (en segundos) para repetir el sonido que emite este enemigo.
Poner 0 si se desea que este enemigo no emita el sonido que tiene definido.
Los sonidos que emiten los/las enemigos/as se definen en la entidad 'ConfigSonidosEne'.

DistanciaMin
Distancia mínima que mantendrá este enemigo entre él y el jugador.

AnimacionTipo
Con 'False' el enemigo utilizará la animación 'Run' por lo que estará siempre corriendo.
Con 'True' el enemigo utilizará la animación 'Walk' por lo que estará siempre andando.
Para que estas animaciones queden bien tienes que modificar también el parámetro 'Velocidad' de esta misma entidad, poniendo 3700 para correr o 2200 para andar.

Bloqueo
Con valor 'True', el jugador colisionará con este enemigo y sufrirá un ligero retroceso.
Con valor 'False', el jugador atravesará este enemigo.

Comandos_colision
Introduciremos el nombre del fichero de comandos .ini que contiene los comandos que deseamos procesar cuando el enemigo colisione con el jugador. Este fichero .ini ha de estar en la carpeta 'Comandos' de 'Entidad 3D'.
Procesar comandos cuando el jugador colisiona con un enemigo en concreto nos puede servir por ejemplo para presentar diálogos al jugador, darle objetos, pedirle algo, reproducir una voz, darle una password, etc..

Comandos_impacto
Introduciremos el nombre del fichero de comandos .INI que contiene los comandos que deseamos procesar cuando el enemigo reciba un impacto de arma del jugador. Este fichero .INI ha de estar en la carpeta 'Comandos' de 'Entidad 3D'.
Procesar comandos cuando el enemigo recibe un impacto, combinado con 'False' en el parámetro 'Pierde_salud', nos puede servir por ejemplo para controlar manualmente (mediante comandos y condiciones) la pérdida de salud de enemigos concretos, así como establecer en que momento debe morir el enemigo.
Si este fichero .INI contiene alguna condición 'IF', el resultado de la primera condición que se encuentre en el fichero .INI se tomará para decidir matar al enemigo. Si el fichero de comandos .INI no contiene ninguna condición, el enemigo morirá al instante.

Pierde_salud
Con valor 'True', el enemigo perderá salud por impactos de armas del jugador.
Con valor 'False', el enemigo no perderá salud por impactos de armas del jugador.
En cualquiera de los dos casos, al recibir un impacto se producirá la explosión de sangre (si está activada), el sonido de daño y todo lo relacionado cuando es alcanzado un enemigo, pero en el caso de que tengamos establecido como 'False' este parámetro no se restará salud de forma automática.
No confundir este parámetro con el parámetro 'Inmune' que permite evitar que el enemigo sufra daños en función de la arma que le impacta y sin producir ningún efecto en el enemigo.

Scale
Permite agrandar o reducir el tamaño del enemigo/personaje, siendo el valor 2.7 para definir un tamaño normal si utilizamos personajes del tipo Half Life o Counter Strike.

Inmune
Mediante un valor de 0 a 4, permite establecer el nivel de invulnerabilidad del enemigo/personaje frente a las armas del jugador:
Con valor **0** no es inmune a ninguna arma del jugador.
Con valor **1** es sólo inmune al arma número 1 del jugador.
Con valor **2** es inmune a las armas número 1 y 2 del jugador.
Con valor **3** es inmune a las armas número 1, 2 y 3 del jugador.
Con valor **4** es inmune a las armas número 1, 2, 3 y 4 del jugador, es decir, a todas las armas.
Usa el valor 4 si quieres que este enemigo/personaje no lo pueda matar el jugador por la razón que sea, por ejemplo que sea un personaje vital para el juego que espera que se le entregue algo para seguir el desarrollo de la aventura.

Destino1 Destino2 Destino3 Destino4 Destino5
Estos 5 parámetros permiten seleccionar entidades 'Posicion' para fijar una ruta que seguirá el enemigo/personaje.
El funcionamiento es muy sencillo: Al dar comienzo el nivel el enemigo/personaje se dirigirá a la posición que hayas seleccionado en 'Destino1', cuando llegue a esa posición se dirigirá a la posición seleccionada en 'Destino2' y así hasta 'Destino5', cuando llegue a ese destino o a un destino que hayas dejado como 'null' volverá a comenzar por el 'Destino1'.
Si no quieres que este enemigo/personaje utilice este sistema de rutas debes dejar 'Destino1' como 'null'.
Debes añadir entidades '**Posicion**' a tu nivel para fijar las posiciones que luego te aparecerán al seleccionar los 5 destinos.

Distancia1_disparar
Permite configurar la distancia mínima a la que debe estar el jugador para que este enemigo utilice el arma número 1.

Distancia2_disparar
Permite configurar la distancia mínima a la que debe estar el jugador para que este enemigo utilice el arma número 2.

Distancia3_disparar
Permite configurar la distancia mínima a la que debe estar el jugador para que este enemigo utilice el arma número 3.

Distancia4_disparar
Permite configurar la distancia mínima a la que debe estar el jugador para que este enemigo utilice el arma número 4.
En el caso que el arma número 4 sea cuerpo a cuerpo (puñal, espada, puños, ...) es conveniente fijar una baja distancia (por ejemplo 200) para que el enemigo sólo ataque si está casi al lado del jugador.
También si es una arma cuerpo a cuerpo debes establecer 'True' en el parámetro 'Dispara4_parado', ya que de lo contrario el enemigo no reproducirá su animación 'Shoot' y por lo tanto no veremos como mueve la espada, puños, dagas o lo que hayas establecido como arma cuerpo a cuerpo.
Los parámetros 'Distancia1_disparar', 'Distancia2_disparar', 'Distancia3_disparar' y 'Distancia4_disparar' también se pueden utilizar para crear armas fijas que se activan por la proximidad del jugador.

Dispara1_parado
Con valor 'True', el enemigo (cuando haya decidido disparar al jugador utilizando el arma 1) lo hará completamente detenido y reproduciendo la animación 'Shoot' que tenga asociada el .ACT que representa este enemigo.
Con valor 'False', este enemigo disparará en la situación que esté en ese momento ya sea parado, corriendo o andando. Si está corriendo o andando no se reproducirá la animación 'Shoot' de este enemigo ya que se estará reproduciendo la propia animación de andar o correr.

Dispara2_parado
Igual que el parámetro 'Dispara1_parado' pero utilizando el arma 2.

Dispara3_parado
Igual que el parámetro 'Dispara1_parado' pero utilizando el arma 3.

Dispara4_parado
Igual que el parámetro 'Dispara1_parado' pero utilizando el arma 4.
Si se está utilizando el arma número 4 como arma cuerpo a cuerpo es conveniente poner este parámetro a 'True', de lo contrario cuando el enemigo ataque utilizando el arma 4 no se verá la animación 'Shoot' de este enemigo, que se supone que es una animación de ataque del enemigo con el arma cuerpo a cuerpo (espada, palo, daga, puñal, puños, zarpas, ...), o lo que hayas diseñado para este enemigo.

Mov_lateral_azar

Con valor 'False' permite que cuando los enemigos están en modo ataque persiguiendo al jugador no realicen movimientos laterales aleatorios.
Con valor 'True' los enemigos realizarán movimientos laterales aleatorios en modo ataque.
Si bien estos movimientos laterales hacen al enemigo un blanco más difícil, en ocasiones pueden resultar poco realistas y marear al jugador en exceso.
Aún con 'False' en este parámetro, cuando el enemigo ya esté casi encima del jugador si realizará estos movimientos laterales para no quedarse clavado delante del jugador y ser un blanco seguro.
También seguirá realizando estos movimientos cuando el enemigo no tenga a la vista el jugador y tenga que esquivar obstáculos.

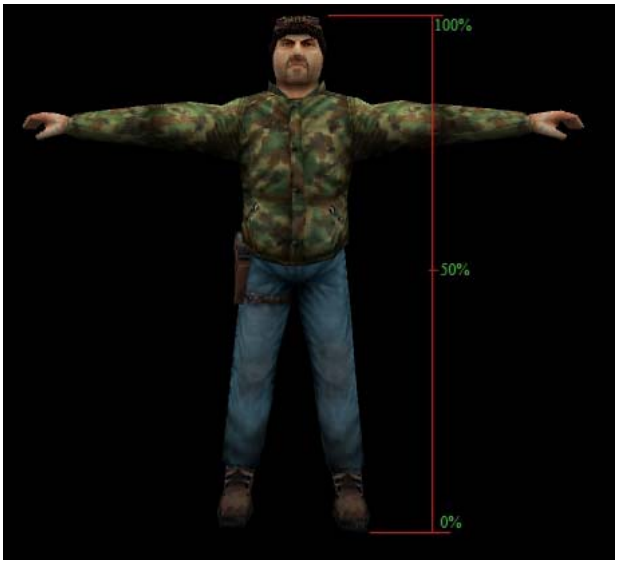
InmuneEnergia

Permite establecer niveles de inmunidad de este enemigo frente a las armas de energía del jugador:
Con valor **-1** es inmune a todas las armas de energía.
Con valor **0** no es inmune a ninguna arma de energía.
Con valores de **1** a **20** es inmune al número de arma de energia que corresponda el valor.

ExplosionMorir

Con valor 'True', cuando el enemigo muera se producirá un efecto de explosión en su posición. Este efecto está definido por las siguientes imágenes:
bmp\explode\2exp01.bmp
bmp\explode\2exp02.bmp
bmp\explode\2exp03.bmp
bmp\explode\2exp04.bmp
bmp\explode\2exp05.bmp
bmp\explode\2exp06.bmp
bmp\explode\a_2exp01.bmp
bmp\explode\a_2exp02.bmp
bmp\explode\a_2exp03.bmp
bmp\explode\a_2exp04.bmp
bmp\explode\a_2exp05.bmp
bmp\explode\a_2exp06.bmp
El efecto de sonido que acompaña a esta explosión está definido por el siguiente fichero:
wav\explosion5.wav

Min_zona_critica y Max_zona_critica



Nos permite definir una zona del enemigo particularmente crítica, donde si es alcanzada por algún disparo del jugador le restará al enemigo una cantidad adicional de salud.
De esta manera podemos hacer, por ejemplo, que la cabeza sea una zona crítica de un enemigo en concreto y que al recibir un impacto, este enemigo muera al acto.
La zona crítica de un enemigo la definen dos porcentajes de la altura del enemigo. Por ejemplo, fíjate en el siguiente enemigo:

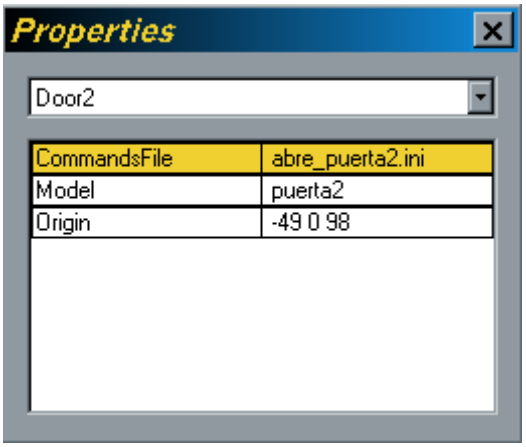
Su porcentaje más bajo del 0% está en los pies y su porcentaje más alto del 100% encima de la cabeza. Si por ejemplo quisiéramos definir su cabeza como zona crítica tendríamos que establecer el valor 80 para el parámetro **Min_zona_critica** y el valor 100 para el parámetro **Max_zona_crítica**:

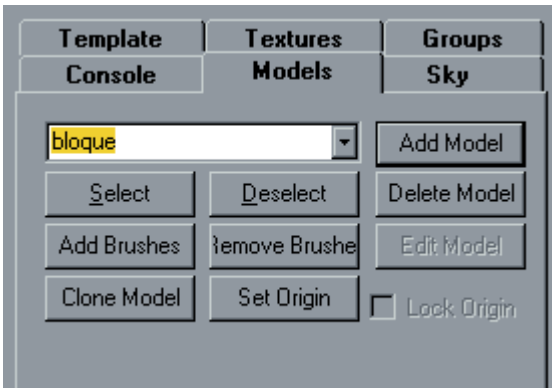
La cantidad de salud adicional que se restará al enemigo si es alcanzado por un disparo en su zona crítica, dependerá del tipo de arma que le haya disparado. Usa las entidades ConfigArmaUno, ConfigArmaTres y ConfigArmaCuatro para indicar esa cantidad de salud en el parámetro '**Resta_salud_critica**'.
Si no quieres definir una zona crítica en el enemigo deja los parámetros **Min_zona_critica** y **Max_zona_critica** con valor 0.

Fíjate que si modificas y combinas los parámetros de esta entidad de forma adecuada, más que enemigos podrás crear personajes normales que interactúan con el jugador y se mueven por el escenario sin ser necesariamente enemigos que haya que matar.
También puedes utilizar esta entidad para fijar armas inmóviles sin animaciones de caminar o andar pero será capaz de disparar apuntando al jugador. Cuando digo sin animaciones de caminar o andar no quiero decir que el .ACT no tenga esos nombres de animación, siempre ha de tener las 9 animaciones obligatorias, por lo menos sus nombres, aunque podemos hacer que las animaciones que no se utilizarán no hagan absolutamente nada o sean repetidas de la animación 'Shoot' por ejemplo.

» Cuando el enemigo no esté ni en modo cobarde ni en modo indiferente, estará en modo ataque.
» Consulta la sección 'Enemigos' para ver un ejemplo de como añadir enemigos al escenario.
» Para que a un enemigo se le vea llevando un arma, el diseño 3D original del enemigo debe tener el arma enganchada en la mano derecha. Consulta el tutorial de 'MilkShape 3D' para realizar esto.
» Los enemigos entre ellos no se causan daños aunque se disparen por accidente.
» Cuando a un enemigo se le acaban las municiones de las armas no tiene forma de recargarlas, a menos que muera y vuelva a reaparecer, tenlo en cuenta a la hora de armar a los enemigos.
» El enemigo decide utilizar una arma u otra en función de la capacidad de fuego de las armas de las que disponga en ese momento, siendo el arma número 4 la primera de su lista, la número 3 la siguiente, la número 2 la siguiente y finalmente la número 1. Pero si hemos fijado unas distancias mínimas para usar las armas también lo tendrá en cuenta y no las utilizará si el jugador no está dentro de la distancia del *arma que tiene establecida, seleccionando en este caso la siguiente mejor.*

Door





Esta entidad se utiliza para definir puertas que pueda abrir el jugador y en ocasiones hasta los propios enemigos.

Parámetros de esta entidad:

CommandsFile
Introduciremos el nombre del fichero de comandos .ini que contiene los comandos que deseamos procesar cuando el jugador toque la puerta. Si este fichero .ini contiene alguna condición 'IF', el resultado de la primera condición que se encuentre en el fichero .ini se tomará para decidir abrir la puerta o no.
Si sólo quieres que la puerta se abra sin más y no quieres asociar ficheros de comandos, deja en blanco la casilla de este parámetro.

Model
Selecciona aquí el 'model' que representa la puerta.
Recuerda que estos 'models' son los que tenemos definidos en la siguiente pestaña de 'World Editor':
Es imprescindible que este parámetro no se quede 'null', de lo contrario tu juego dejará de funcionar de forma brusca.

EnemigoAbre
Con valor 'True', la puerta la podrá abrir un enemigo (o personaje definido con una entidad 'BotActorStart').
Con valor 'False', la puerta no la podrá abrir un enemigo (o personaje definido con una entidad 'BotActorStart').

CommandsFileE
Introduciremos el nombre del fichero de comandos .ini que contiene los comandos que deseamos procesar cuando un enemigo (o personaje definido con una entidad 'BotActorStart') toque la puerta.
Para que se comience a procesar este fichero .ini, el parámetro '**EnemigoAbre**' debe estar a '**True**'.
Si este fichero .ini contiene alguna condición '**IF COND**', el resultado de la primera condición que se encuentre en el fichero .ini se tomará para decidir abrir la puerta o no.
Si sólo quieres que la puerta se abra sin más y no quieres asociar ficheros de comandos, deja en blanco la casilla de este parámetro y a '**True**' el parámetro '**EnemigoAbre**'.
Si quieres que la puerta se abra con las mismas condiciones que el jugador, indica el mismo fichero .ini en los parámetros '**CommandsFileE**' y '**CommandsFile**' , y pon a '**True**' el parámetro '**EnemigoAbre**'.
Ten en cuenta que si un enemigo o personaje abre una puerta que debería estar cerrada para el jugador hasta que éste complete alguna misión, puede alterar el normal desarrollo de los acontecimientos de tu juego.

AbrirPorImpacto
Con valor 'True', la puerta se podrá abrir si el jugador dispara sobre la puerta.
Con valor 'False', la puerta no se podrá abrir si el jugador dispara sobre la puerta.
Sólo funciona con las armas número 1 (Blaster) y número 3 (Lanza-cohetes) del jugador, no de los enemigos.

CommandsFile2
Introduciremos el nombre del fichero de comandos .ini que contiene los comandos que deseamos procesar cuando el jugador dispare sobre la puerta. Este fichero .ini ha de estar en la carpeta 'Comandos' de 'Entidad 3D'.
Para que se comience a procesar este fichero .ini, el parámetro '**AbrirPorImpacto**' debe estar a '**True**'.
Si este fichero .ini contiene alguna condición '**IF COND**', el resultado de la primera condición que se encuentre en el fichero .ini se tomará para decidir abrir la puerta o no.
Sólo funciona con las armas número 1 (Blaster) y número 3 (Lanza-cohetes) del jugador, no de los enemigos.

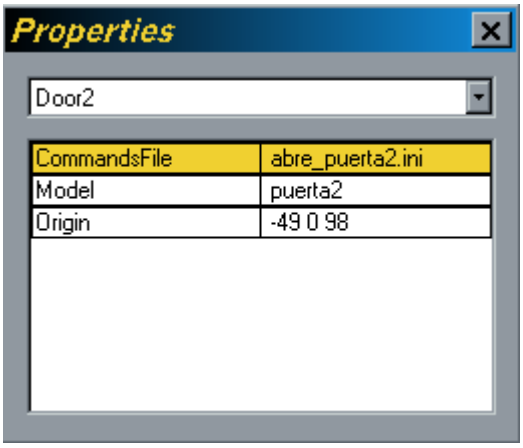
Origin
Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos que normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Boton_accion
Con valor 'True', los comandos contenidos en el fichero especificado en 'CommandsFile' se procesarán si el jugador toca la puerta y al mismo tiempo pulsa el botón o tecla 'Acción'.
Con valor 'False', sólo será necesario que el jugador toque la puerta para procesar los comandos que contiene el fichero .ini que hemos especificado en 'CommandsFile'.

Ciclico
Con valor 'True', la puerta se abrirá y se cerrará sin parar.
Con valor 'False', la puerta se abrirá y luego se cerrará.

Consulta la sección 'Abriendo puertas' para ver ejemplos de como usar esta entidad conjuntamente con ficheros de comandos.

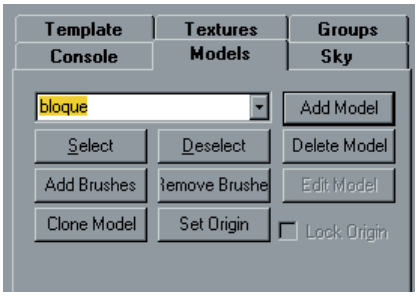
MovingPlat



Esta entidad se utiliza para definir ascensores o plataformas móviles que se activan al paso del jugador o de los enemigos.

Parámetros de esta entidad:

CommandsFile
Introduciremos el nombre del fichero de comandos .ini que contiene los comandos que deseamos procesar cuando el jugador toque la plataforma.
Si este fichero .ini contiene alguna condición 'IF', el resultado de la primera condición que se encuentre en el fichero .ini se tomará para decidir mover o no la plataforma.
Si sólo quieres que la plataforma se mueva sin más y no quieres asociar ficheros de comandos, deja en blanco la casilla de este parámetro.



Model
Selecciona aquí el 'model' que representa la plataforma.
Recuerda que estos 'models' son los que tenemos definidos en la siguiente pestaña de 'World Editor':
Es imprescindible que este parámetro no se quede 'null', de lo contrario tu juego dejará de funcionar de forma brusca.

EnemigoMueve
Con valor 'True', la plataforma la podrá mover un enemigo (o personaje definido con una entidad 'BotActorStart').
Con valor 'False', la plataforma no la podrá mover un enemigo (o personaje definido con una entidad 'BotActorStart').

CommandsFileE
Introduciremos el nombre del fichero de comandos .ini que contiene los comandos que deseamos procesar cuando un enemigo (o personaje definido con una entidad 'BotActorStart') toque la plataforma.
Para que se comience a procesar este fichero .ini, el parámetro '**EnemigoMueve**' debe estar a '**True**'.
Si este fichero .ini contiene alguna condición '**IF COND**', el resultado de la primera condición que se encuentre en el fichero .ini se tomará para decidir mover la plataforma o no.
Si sólo quieres que la plataforma se mueva sin más y no quieres asociar ficheros de comandos, deja en blanco la casilla de este parámetro y a '**True**' el parámetro '**EnemigoMueve**'.
Si quieres que la plataforma se mueva con las mismas condiciones que el jugador, indica el mismo fichero .ini en los parámetros '**CommandsFileE**' y '**CommandsFile**' , y pon a '**True**' el parámetro '**EnemigoMueve**'.
Ten en cuenta que si un enemigo o personaje mueve una plataforma que debería estar inmóvil para el jugador hasta que éste complete alguna misión, puede alterar el normal desarrollo de los acontecimientos de tu juego.

MoverPorImpacto
Con valor 'True', la plataforma se podrá mover si el jugador dispara sobre ella.
Con valor 'False', la plataforma no se podrá mover si el jugador dispara sobre ella.
Sólo funciona con las armas número 1 (Blaster) y número 3 (Lanza-cohetes) del jugador, no de los enemigos.

CommandsFile2
Introduciremos el nombre del fichero de comandos .ini que contiene los comandos que deseamos procesar cuando el jugador dispare sobre la plataforma. Este fichero .ini ha de estar en la carpeta 'Comandos' de 'Entidad 3D'.
Para que se comience a procesar este fichero .ini, el parámetro '**MoverPorImpacto**' debe estar a '**True**'.
Si este fichero .ini contiene alguna condición '**IF COND**', el resultado de la primera condición que se encuentre en el fichero .ini se tomará para decidir mover la plataforma o no.
Sólo funciona con las armas número 1 (Blaster) y número 3 (Lanza-cohetes) del jugador, no de los enemigos.

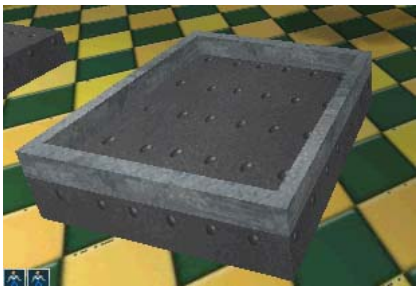
Origin
Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Boton_accion
Con valor 'True', los comandos contenidos en el fichero especificado en 'CommandsFile' se procesarán si el jugador toca la plataforma y al mismo tiempo pulsa el botón o tecla 'Acción'.
Con valor 'False', sólo será necesario que el jugador toque la plataforma para procesar los comandos que contiene el fichero .ini que hemos especificado en 'CommandsFile'.

Ciclico
Con valor 'True', la plataforma se moverá continuamente.
Con valor 'False', la plataforma realizará su recorrido previsto y luego se detendrá.

Un consejo: Si tienes una plataforma que se mueve horizontalmente y quieres que al subirse el jugador en ella lo transporte, no realices una plataforma plana como esta:

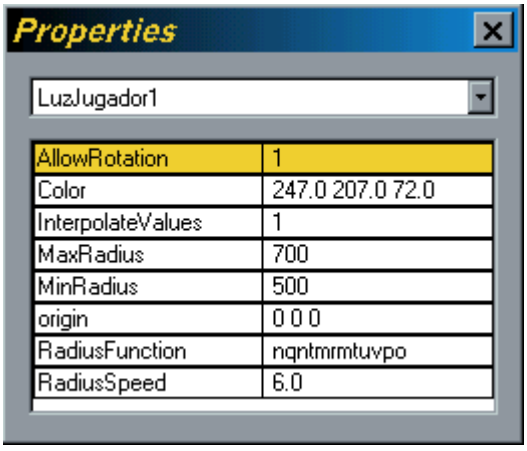
... sino como esta:



Consulta la sección 'Abriendo puertas' para ver ejemplos de como usar la entidad 'Door' que funcionalmente es idéntica a 'MovingPlat'.

LuzJugador

Esta entidad se utiliza para definir una fuente de luz que pueda llevar el jugador.



Parámetros de esta entidad:

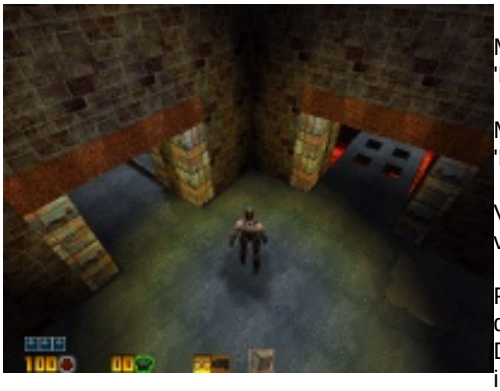
Color
Permite seleccionar el color de la luz.

Origin
Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

AllowRotation
'No se utiliza en esta entidad'.

InterpolateValues
Se utiliza para producir suaves variaciones en la luz. Prueba valores del 0 al 3.

RadiusFunction
Aquí entraremos unas serie de letras que determinarán la secuencia y niveles de variación de la luz. La letra 'a' es para muy oscuro, mientras que la letra 'z' es para muy brillante.



MaxRadius
Máximo alcance de la luz cuando se utilizan variaciones de luz con valores en 'RadiusFunction'.

MinRadius
Mínimo alcance de la luz cuando se utilizan variaciones de luz con valores en 'RadiusFunction'.

RadiusSpeed
Velocidad en las variaciones de la luz cuando se utilizan variaciones de luz con valores en 'RadiusFunction'.

Arc
Permite definir el ángulo de apertura de la luz en el caso de que queramos definir una luz tipo linterna para el jugador.
Dejar este valor a 0 si lo que se quiere definir es una luz tipo antorcha que ilumina hacia todos los lados.

Ejemplo 1: Para conseguir el efecto de luz de una antorcha utilizaremos estos valores:

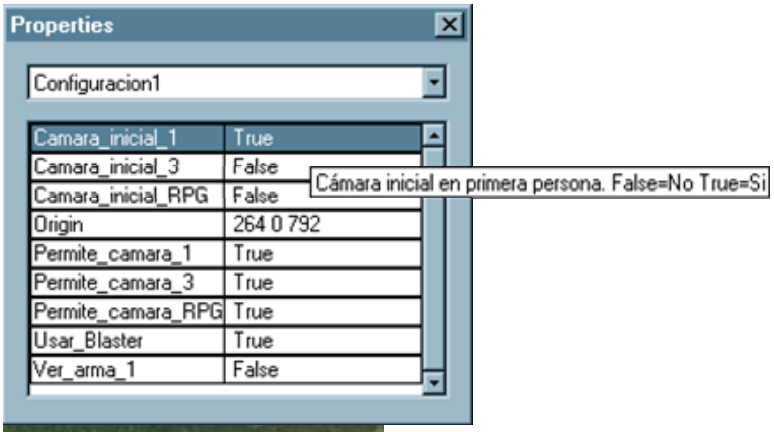
Color: Selecciona un color naranja
MaxRadius: 500
MinRadius: 50
RadiusFunction: gkgtemrhmth
RadiusSpeed: 6.7
Arc: 0

Ejemplo 2: Para conseguir el efecto de luz tipo linterna utilizaremos estos valores:
Color: Selecciona un color amarillo claro
MaxRadius: 800
MinRadius: 700
RadiusFunction: m
RadiusSpeed: 6.0
Arc: 60

Es necesaria una entidad 'LuzJugador' por nivel y la puedes colocar donde quieras mientras esté en el interior del mapa.

Consulta la entidad 'ItemObject' para saber como crear objetos recogibles por el jugador y que sean fuentes de luz.

Configuracion



Esta entidad establece distintos parámetros generales que afectarán a todo el juego.

Parámetros de esta entidad:

Origin
Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Camara_inicial_1
Con valor 'True' hacemos que el juego comience con la vista en primera persona (desde los ojos del jugador):

Camara_inicial_3
Con valor 'True' hacemos que el juego comience con la vista en tercera persona (siempre desde detrás del jugador):

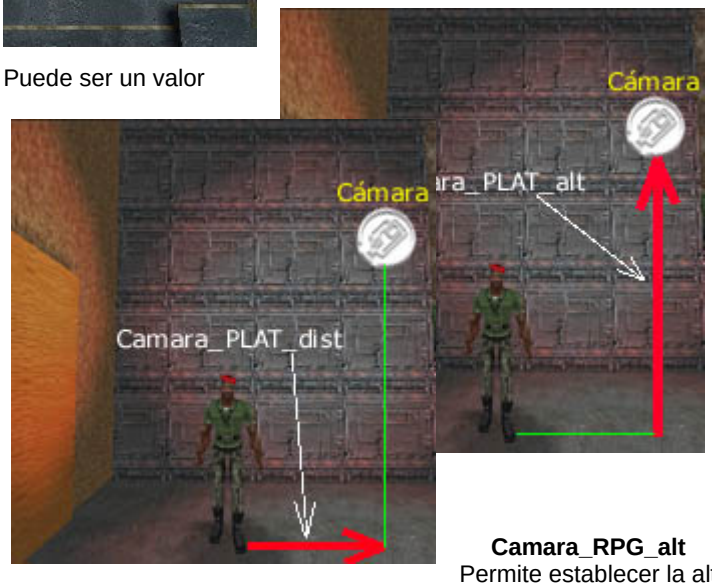
Nota: Esta cámara inteligente tiene un comportamiento muy similar a la utilizada en juegos como Tomb Raider III, que es una cámara que sigue al jugador ligeramente por detrás y cuando la cámara colisiona con bloques del escenario esta se acerca al jugador intentando en todo momento no colisionar con nada.

Camara_inicial_RPG
Con valor 'True' hacemos que el juego comience con la vista tipo RPG (siempre desde detrás y arriba del jugador):

Nota: El punto de vista de esta cámara no da buenos resultados en todos los escenarios, pero si quieres realizar un juego con esta vista lo ideal es diseñar el escenario adecuado a esta cámara, teniendo en cuenta que las paredes sean siempre más bajas que la altura de la cámara y las habitaciones sin techo para poder ver su interior desde arriba.
Ten en cuenta que cuando esta cámara colisione con bloques del escenario (paredes, techos, suelos, etc..) la cámara atravesará esos bloques pero en todo momento seguirá mostrando todo el escenario, personajes y objetos de forma correcta.

Camara_inicial_PLAT
Con valor 'True' hacemos que el juego comience con la vista tipo Plataformas (sin ninguna rotación de cámara, el personaje del jugador se podrá ver desde cualquier ángulo según los giros que realice el jugador):
Nota: Esta cámara no da buenos resultados en todos los escenarios, pero si quieres realizar un juego con esta vista lo ideal es diseñar el escenario adecuado a esta cámara (normalmente juegos de plataformas 3D horizontales, isométricos, con vista aérea, estilo RPG, ...), teniendo en cuenta que la cámara no toque por detrás de ella ni por encima con los límites del escenario, limitando para ello la zona de actuación del jugador. También es conveniente que las paredes del interior del escenario sean siempre más bajas que la altura de la cámara si no quieres ver como la cámara las atraviesa, aunque si lo hace seguirá mostrando en todo momento de forma correcta el escenario.

Puede ser un valor



Camara_PLAT_alt
Permite establecer la altura de la cámara Plataformas. entre 10 y 1000.

Camara_PLAT_dist
Permite establecer la distancia de la cámara Plataformas. Puede ser un valor entre 10 y 1000.

Camara_RPG_alt
Permite establecer la altura de la cámara RPG. Puede ser un valor entre 10 y 1000.

Permite_camara_1
Con valor 'True' el jugador podrá activar la vista en primera persona pulsando la tecla 'Q'.
Con valor 'False' no se le permite al jugador activar la vista en primera persona.

Permite_camara_3
Con valor 'True' el jugador podrá activar la vista en tercera persona pulsando la tecla 'A'.
Con valor 'False' no se le permite al jugador activar la vista en tercera persona.

Permite_camara_RPG
Con valor 'True' el jugador podrá activar la vista tipo RPG pulsando la tecla 'Z'.
Con valor 'False' no se le permite al jugador activar la vista tipo RPG.

Permite_camara_PLAT
Con valor 'True' el jugador podrá activar la vista tipo Plataformas pulsando la tecla 'X'.
Con valor 'False' no se le permite al jugador activar este tipo de cámara.

Utilizar_Joystick
Con valor 'True' el jugador podrá utilizar el Joystick o Gamepad para jugar.
Con valor 'False' el jugador no podrá utilizar el Joystick o Gamepad para jugar.
Nota: Con esta opción activada y una mala calibración del Joystick o Gamepad podría causar movimientos en el personaje sin que el jugador intervenga.

VolumenMusicaFondo
Permite establecer el volumen de sonido de la música de fondo que suena mientras se está jugando. Acepta valores del 0 al 100.

Ver_Sombras
Con 'True', se verá una especie de sombra debajo de cada objeto o personaje.

Lava_salud
Puede ser un valor entre 0 y 255.
Permite establecer la cantidad de salud que se resta al jugador al tocar la lava del escenario.

FicheroComandos
Nombre de un fichero de comandos .INI que se lanzará al dar comienzo el primer nivel del juego.
Este fichero .INI debe estar en la carpeta '**comandos**' de Entidad 3D.
Esto te permitirá ejecutar comandos justo al inicio del juego, con la posibilidad de por ejemplo cargar variables, presentar mensajes, presentar desde el principio un marcador personalizado, un mensaje en audio que sólo aparezca al principio, etc...
En definitiva toda la potencia de los comandos y condiciones desde el mismo instante que de comienzo el juego y sin que el jugador tenga que hacer nada.
Recuerda que dispones de la posibilidad de lanzar ficheros de comandos .INI al dar comienzo cada nivel utilizando el parámetro '**CommandsFile**' de la entidad 'ChangeLevel'.

FicheroAcciones
Nombre de un fichero de comandos .INI que se procesará cada vez que el jugador use una acción del menú de acciones.
De esta forma podemos controlar las acciones que realiza o intenta realizar el jugador en cualquier lugar y momento del juego.

UsarFicherosEnc
Con 'True' el programa no buscará, en la carpeta '**comandos**' los ficheros **.ini** ni **.dlg** para procesar los comandos o textos, aunque en su lugar buscará sus equivalentes encriptados cuya extensión acaba en **.ini.enc** y **.dlg.enc**.
Por ejemplo: si en nuestro juego hacemos referencia en alguna entidad a un fichero '**puerta.ini**', Entidad 3D no usará (aunque exista) ese fichero, sino que buscará '**puerta.ini.enc**' para procesar sus comandos.
En el caso de los ficheros **.dlg** sucede exactamente lo mismo, el programa buscara por ejemplo '**midialogo.dlg.enc**' en lugar de '**midialogo.dlg**'.
Con 'False' en este parámetro el programa usará los ficheros **.ini** y **.dlg** de forma normal.
Consulta el anexo 'Encriptación de datos' para saber como se crean los ficheros '**.enc**' y para que sirven.

UsarFicherosEnc
¿ Otra vez este parámetro ? Si, este parámetro tiene una segunda función.
Con 'False' se permite al jugador usar los comandos de consola para hacer trampas en el juego.
Con 'True' no se permite al jugador hacer trampas usando los comandos de la consola.

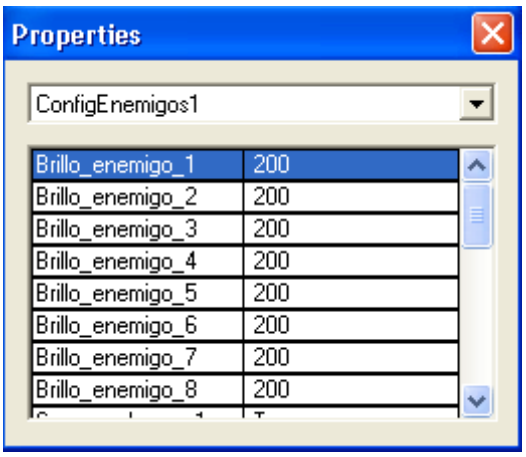
SombrasDinMax
De 0 a 3.
Permite indicar el número máximo de sombras de tipo 2 (sombras dinámicas) que se calcularán al mismo tiempo para cada objeto o personaje 3D.
Dependiendo de la cantidad de luces que incidan en un objeto se proyectarán diversas sombras simultáneamente, pero el cálculo de todas las sombras para todos los objetos puede sobrecargar en exceso el motor 3D. Mediante el parámetro 'SombrasDinMax' podremos modificar la cantidad de sombras proyectadas.
Normalmente, con valor **1** es suficiente, aunque con valor **2** podremos conseguir transiciones de una sombra a otra más suaves y hacer que aparezcan más sombras. Es cuestión de experimentar y observar lo que se adapte mejor a la disposición particular de luces de tu mapa.

Con valor **1** se calculará una sola sombra.

Con valor **2** se calcularán dos sombras.

***Nota**: Sólo es necesaria una entidad 'Configuracion' para todo el juego, con poner esta entidad en el primer nivel será suficiente. Esta entidad la puedes colocar donde quieras mientras se encuentre en el interior del mapa.*

ConfigEnemigos



Esta entidad establece distintos parámetros generales que afectarán a todos los enemigos que hayamos situado en nuestro nivel con la entidad 'BotActorStart'.

Parámetros de esta entidad:

Brillo_enemigo_1

Con un valor que puede variar entre 0 y 255, podemos dar más o menos brillo al enemigo número 1 en nuestro juego.
Hay que tener en cuenta que este nivel de brillo se mantendrá para todos los niveles que compongan el juego y para este número de enemigo.
Existen parámetros 'Brillo_enemigo_' para los ocho tipos de enemigos.

Sangre_charco_1

Con 'True' aparecerá un charco de sangre debajo del enemigo número 1 cuando esté muerto.
Con 'False' no aparecerá ningún charco de sangre debajo del enemigo número 1 cuando esté muerto.
Existen parámetros 'Sangre_charco_' para los ocho tipos de enemigos.

Transparencia_1

De 0 a 255.
Nivel de transparencia del enemigo número 1.
255 opaco, 0 completamente transparente, 128 semi-transparente.
Existen parámetros 'Transparencia_' para los ocho tipos de enemigos.

Sombra_1

De 0 a 2.
Tipo de sombra para el enemigo número 1.
Existen parámetros 'Sombra_' para los ocho tipos de enemigos.

Valor **0** si quieres que el enemigo no tenga sombra.

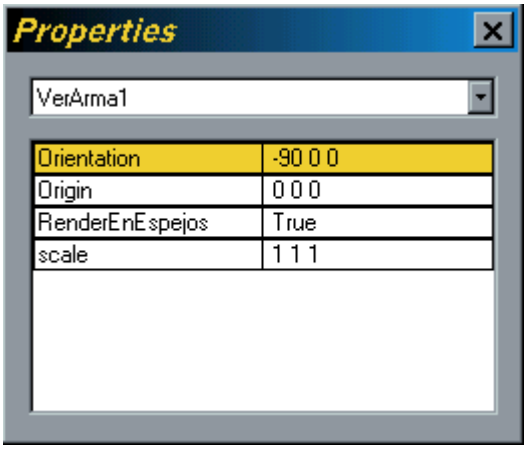
Valor **1** para una sombra en círculo.

Valor **2** para una sombra dinámica realista.

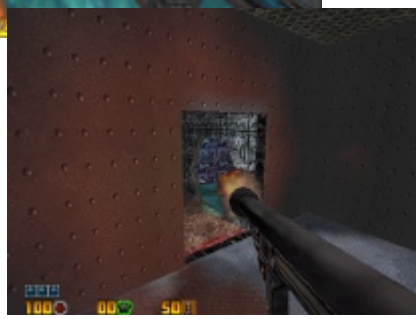
Evidentemente la sombra número 2 es la que consume más recursos del motor 3D, por lo que a tú elección queda equilibrar el uso de este tipo de sombras en tu juego.

» La configuración que establezcas en una entidad 'ConfigEnemigos' de un nivel, tendrá efecto en los niveles siguientes a menos que añadas otra entidad 'ConfigEnemigos' en niveles posteriores.
» Esta entidad la puedes colocar donde quieras mientras se encuentre en el interior del mapa.

VerArma



Esta entidad se utiliza para permitir que el jugador pueda ver, en la vista en primera persona, el arma que está utilizando.



Parámetros de esta entidad:

Orientation

No es necesario modificar este valor.

Origin

Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

RenderEnEspejos

Con 'True', el arma del jugador se verá reflejada en los espejos del escenario.
Con 'False', el arma del jugador no se verá reflejada en los espejos del escenario.

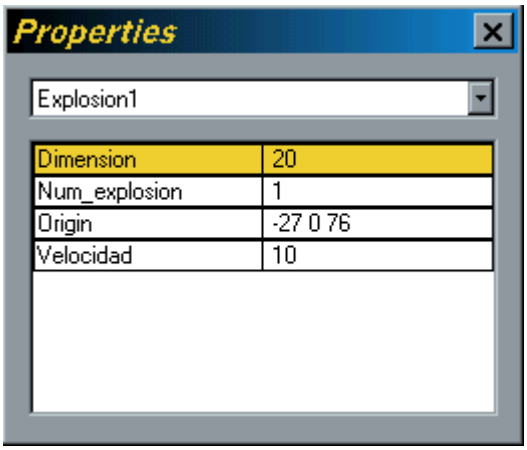
Scale

No es necesario modificar este valor.

Sólo es necesaria una entidad 'VerArma' para todo el juego, con poner esta entidad en el primer nivel será suficiente. Esta entidad la puedes colocar donde quieras mientras se encuentre en el interior del mapa.

Consulta la entidad 'DeathMatchStart' para saber como activar y desactivar la vista del arma.

Explosion



Esta entidad nos permite colocar explosiones en puntos concretos del escenario y que podremos activar mediante el comando 'EXPL' en el momento que queramos durante el transcurso del juego.

Parámetros de esta entidad:

Origin

Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Num_explosion
Número de explosión (entre 1 y 255).
Son números que deberemos asignar nosotros manualmente.
Este número será el que se utilizará en el comando 'EXPL' para referirse a una explosión en concreto.
Se puede utilizar el mismo número en diferentes entidades 'Explosion' si lo que queremos es que se activen diferentes explosiones al mismo tiempo y utilizando un solo comando 'EXPL'.

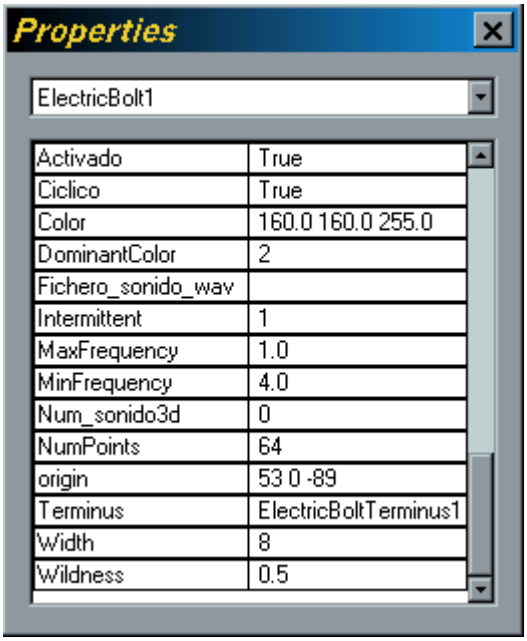
Dimension
Tamaño que tendrá esta explosión. A más valor mayor tamaño.

Velocidad
Velocidad de la animación que representará la explosión. A menos valor explosión más lenta.

Puedes poner tantas entidades 'Explosion' como quieras.
Asígnales un número de explosión 'Num_explosion' distinto a cada una si las vas a activar por separado.
Al producirse la explosión se reproducirá automáticamente un efecto de sonido de explosión.
Esta entidad se puede utilizar para por ejemplo:
Provocar una explosión en una puerta donde el jugador ha colocado un explosivo retardado.
Provocar cuatro explosiones simultáneas para volar una pared.
Provocar una explosión en un laboratorio para eliminar un cultivo letal de virus.
Provocar miniexplosiones en un panel que acaba de dejar de funcionar.
Hay que entender que este efecto hace solamente eso, el efecto, ni te va a destruir una puerta, ni una trampilla ni por supuesto eliminar a los enemigos, es sólo un efecto de explosión en los lugares que has situado las entidades 'Explosion' para que te permita dar más realismo a las posibles acciones que definas con otros comandos de Entidad 3D.
Este tipo de efecto es ideal si deseas cambiar o eliminar algún elemento del escenario durante el transcurso del juego y quieres dar el efecto de que lo ha provocado una explosión. Tú definirás que condiciones y comandos procesas para llevar a cabo esas eliminaciones o cambios, pero si justo antes de eso provocas una gran explosión en el lugar del cambio o eliminación, el efecto puede resultar de lo más realista.

Consulta la sección del comando 'EXPL' para ver un ejemplo de como activar entidades 'Explosion' durante el transcurso del juego.

ElectricBolt



Esta entidad se utiliza para dos funciones distintas en Entidad 3D.
La primera es para crear en el escenario un efecto de rayos de energía con sonido incluido, la explicación de esta entidad para esta función la podrás encontrar en el tutorial de 'World Editor' sección 'Efectos de luz, rayos, niebla y cielo'.
La otra función de esta entidad es la de colocar en el escenario sonidos 3D posicionales. El jugador oirá estos sonidos en el lado del altavoz que se encuentre el sonido en ese momento y con la atenuación correspondiente según la distancia del jugador.

Parámetros de esta entidad que nos interesan para la creación de sonidos 3D:

Origin
Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Num_sonido3d
Número de sonido 3D (entre 1 y 255).
Son números que deberemos asignar nosotros manualmente.
Este número será el que se utilizará en los comandos 'SON0' y 'SON1' para controlar posteriormente este sonido.

Fichero_sonido_wav
Nombre del fichero de sonido (exclusivamente en formato .WAV) que se interpretará.
Este fichero ha de estar en la carpeta 'wav'.

Activado
Con valor 'True' este sonido se comenzará a interpretar desde que de comienzo el nivel.
Con valor 'False' este sonido no comenzará a interpretarse al dar comienzo el nivel.

Ciclico
Con valor 'True' el sonido se interpretará continuamente.
Con valor 'False' el sonido se interpretará una sola vez y sólo al utilizar el comando SON1.

Intermittent
Debemos poner este valor siempre a 0.

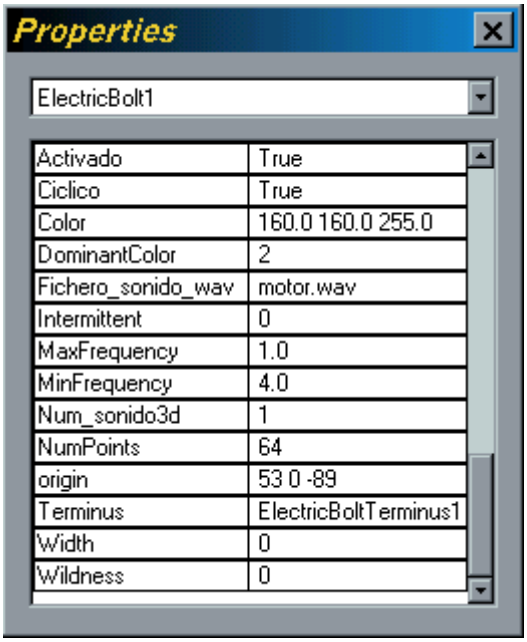
Width
Debemos poner este valor siempre a 0.

Wildness

Debemos poner este valor siempre a 0.

Terminus

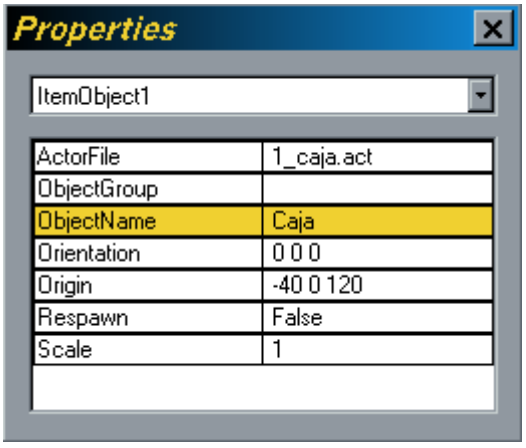
Siempre que añadamos una entidad 'ElectricBolt' a un escenario deberemos añadir obligatoriamente una entidad llamada 'ElectricBoltTerminus' que le acompañará. Pues bien, es en esta casilla llamada 'Terminus' donde debemos seleccionar la entidad 'ElectricBoltTerminus' que acabamos de añadir.
Esta entidad 'ElectricBoltTerminus' sólo se utiliza para fijar una posición que junto con la posición de la entidad 'ElectricBolt' a la que está asociada, fijar una distancia que establezca el tamaño de la fuente del sonido.



Recuerda que durante el transcurso del juego disponemos de los comandos 'SON1' y 'SON0' para activar y desactivar los sonidos 3D, con lo que podremos detener y reanudar la interpretación de estos sonidos según el desarrollo de nuestro juego.
Los ficheros en formato .WAV que se utilizan para este efecto, son ficheros normales .WAV sin nada especial en 3D, ni Sourround ni nada por el estilo, pero puestos en el escenario con la entidad 'ElectricBolt', Entidad 3D los procesa de forma independiente y realizando las variaciones necesarias para dar la sensación de estar colocados en un punto en concreto del escenario.
Ejemplo: Una entidad 'ElectricBolt' para posicionar el sonido 3D 'motor.wav', tendría los siguientes valores:

Consulta las secciones de los comandos 'SON1' y 'SON0' para ver como activar y desactivar sonidos 3D durante el transcurso del juego.

ItemObject



Esta entidad se utiliza para situar todo tipo de objetos en el escenario.

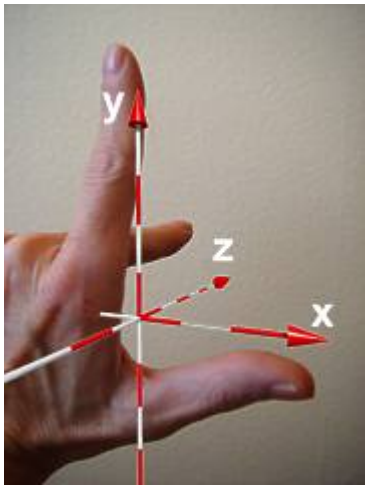
Parámetros de esta entidad:

ActorFile

Introducir aquí el nombre del fichero .act que quieras situar como objeto en el escenario. Este fichero .act ha de estar en la carpeta 'Objects' de 'Entidad 3D'.

ObjectGroup

Podrás asociar distintos objetos a un nombre de grupo para que el jugador sólo pueda llevar en su inventario un solo objeto de cada grupo. Esta caja también se utiliza para asignar a un objeto la propiedad de generar luz si lo lleva el jugador. Basta con rellenar esta caja con la palabra **Luz** y añadir la entidad 'LuzJugador' a nuestro mapa, cuando el jugador recoja este objeto podrá activar y desactivar (con la teclas 'M' y 'N') una luz que siempre le acompañará hasta que pierda el objeto.



ObjectName

Permite asignarle un nombre al objeto. Este nombre sólo lo utiliza 'Entidad 3D' para ponerlo en pantalla mientras el jugador recoge este objeto. Si ponemos más de un objeto igual en nuestro escenario (por ejemplo dos llaves) este parámetro sólo hay que rellenarlo en la primera entidad que hemos añadido para el primer objeto, el resto de objetos de ese tipo tomarán el mismo nombre.

Orientation

Son tres cifras (X Y Z) que determinan la orientación que presentará el objeto en el escenario. Variando estos tres ángulos (con valores entre 0 y 365 grados) podremos hacer aparecer nuestro objeto boca abajo, tumbado, ladeado, etc..
Si ponemos 0 0 0 el objeto se presentará en la orientación inicial de diseño de este objeto 3D.

Ejes X Y Z en Entidad 3D

Origin
Son tres cifras (X Y Z) que determinan la posición de este objeto en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos este objeto (con el ratón) desde 'World Editor'.

Scale
Permite agrandar o reducir el tamaño inicial de nuestro objeto. Si ponemos 1 el objeto aparecerá en el escenario en el tamaño inicial de diseño de este objeto 3D. Si ponemos 2 aparecerá el doble de grande. Si ponemos 0.5 aparecerá a la mitad de su tamaño.

Rotate
Con 'True' haremos que el objeto aparezca rotando sobre si mismo.

Respawn
Con 'True' haremos que el objeto vuelva a aparecer al cabo de unos segundos de haberlo recogido el jugador.

Catch
Con 'False', aún siendo un objeto recogible, podemos hacer que este objeto en concreto no lo sea.

Brillo
Con un valor que puede variar entre 0 y 255 podemos dar más o menos brillo al modelo 3D que represente este objeto.

CommandsFile
Introduciremos el nombre del fichero de comandos .ini que contiene los comandos que deseamos procesar cuando el jugador recoja este objeto. Este fichero .ini ha de estar en la carpeta 'Comandos' de 'Entidad 3D'.

Inventario_objeto
Con 'True', el objeto pasará al inventario de objetos del jugador cuando éste simplemente lo toque.
Con 'False', el objeto no pasará al inventario de objetos del jugador al tocarlo, aunque desaparecerá del escenario.
¿ Para que sirve coger objetos que no se añaden al inventario ?. Pues puede que te interese recoger ciertos objetos que no tengan repercusión en el inventario del jugador pero quieres que cuando el jugador los recoja sólo desaparezcan y se lance un fichero de comandos (el que se indique en el parámetro 'CommandsFile') para controlar lo que quieras al recoger este objeto.
Ejemplos de objetos que se recogen pero que no se añaden al inventario:
- Las bolas del juego 'Pacman'.
- Hechizos que transforman o dan al instante una cualidad especial al jugador.
- Mapas que sólo pueden verse un instante.
- Antídotos contra venenos.
- Depósitos de oxígeno para que el jugador pueda seguir respirando en una cueva.
- Objetos que restan salud al jugador, tendrá que esquivarlos entre otros que le dan salud.

Tiempo_reaparicion
Una vez recogido este objeto, establece el tiempo (en segundos) que ha de transcurrir para que el objeto vuelva a reaparecer.
El parámetro 'Respawn' ha de estar a 'True' para que funcione este tiempo.

Transparencia
De 0 a 255.
Nivel de transparencia del objeto.
255 opaco, 0 completamente transparente, 128 semi-transparente.

Usos
Se usa para asignar la cantidad de usos que el jugador podrá realizar con este objeto. También se puede usar como desgaste para ciertos objetos.
En este parámetro debemos introducir la cantidad de usos máximo que tendrá este objeto cuando entre en el inventario del jugador. Esta cifra la verá el jugador en el interior del icono del objeto en el propio inventario de objetos.
El valor de 'Usos' puede ser de 1 a 99.
Si no quieres que un objeto tenga número de usos, simplemente deja este parámetro con valor 1.

Boton_accion
Con 'True', condiciona la recogida del objeto y el procesamiento del fichero de comandos .INI del parámetro 'CommandsFile', a la pulsación de la tecla 'Acción' por parte del jugador.
Es importante que lo pongas a '**True**' si es un objeto que debe comprar el jugador para conseguirlo.
Con 'False', el objeto será recogido por el jugador sin necesidad de pulsar la tecla 'Acción'. El fichero de comandos .INI que hayas indicado en el parámetro 'CommandsFile' también se procesará sin necesidad de que el jugador use la tecla 'Acción'.

Descrip1
Primera línea de texto para describir este objeto. Máximo 28 caracteres por línea.
Las cuatro líneas de texto (**Descrip1, Descrip2, Descrip3 y Descrip4**) se mostrarán en el cuadro de información del inventario del jugador cuando seleccione este objeto.
También se muestran en el **panel de objeto** que aparece en pantalla cuando el jugador toca la entidad 'ItemObject' de este objeto.

Descrip2
Segunda línea de texto para describir este objeto. Máximo 28 caracteres por línea.

Descrip3
Tercera línea de texto para describir este objeto. Máximo 28 caracteres por línea.

Descrip4
Cuarta línea de texto para describir este objeto. Máximo 28 caracteres por línea.

PanelObjeto
Con 'True', cuando el jugador toque este objeto, el programa presentará el **panel de objeto** conteniendo datos del objeto.
Con 'False', no se mostrará el **panel de objeto** cuando el jugador toque el objeto.

Panel de objeto

Precio_compra
Precio que se establece para este objeto si lo que quieres es que el jugador lo tenga que comprar para conseguirlo.
Este precio se muestra en el **panel de objeto** que aparece en pantalla cuando el jugador toca la entidad 'ItemObject' de este objeto.
Poner **0** si el objeto lo puede recoger el jugador sin necesidad de comprarlo.
Si has puesto un precio debes poner a '**True**' el parámetro '**Boton_accion**' para que funcione la compra de este objeto por parte del jugador.
Para que el jugador pueda comprar este objeto, sólo debe colocarse cerca del objeto y seleccionar la acción '**Comprar**' del menú de acciones, si tiene suficiente dinero la compra se realizará de forma automática, en caso contrarío se informará al jugador de que no tiene suficiente dinero.

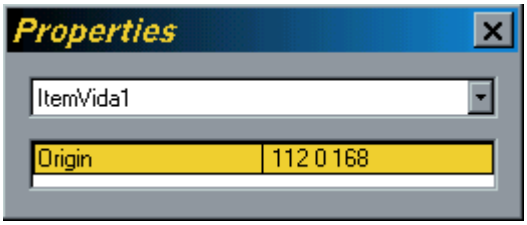
Precio_venta
Precio que se establece para este objeto en el caso que el jugador lo tenga en su inventario de objetos y desee venderlo.
Poner **0** si el objeto no tiene un precio concreto y por lo tanto el jugador no podrá venderlo usando la acción '**Vender**' del menú de acciones.
Este precio aparecerá en el **panel de descripción de objeto** que acompaña al inventario de objetos.
Consulta el comando 'QINVVENDER' para saber como el jugador puede vender objetos.

Sombra
De 0 a 2.
Establece el tipo de sombra para este objeto.
Valor **0** sin sombra.
Valor **1** con sombra en círculo.
Valor **2** con sombra dinámica.
Si quieres ver ejemplos de estos tipos de sombras, consulta la sección de la entidad ConfigEnemigos.

Las imágenes que forman el panel de objeto son:
bmp \ menu \ panel_objeto.bmp
bmp \ menu \ a_panel_objeto.bmp

» Sólo se permiten hasta un máximo de 50 entidades 'ItemObject' por nivel.
» Consulta los comandos 'QINV', 'QTIN' y 'QINVUSO', para saber como eliminar un objeto del inventario del jugador.
» Consulta la sección 'Objetos' para ver ejemplos de esta entidad.

ItemVida



Esta entidad hace aparecer un objeto en el escenario, que al ser recogido por el jugador le representará una vida extra.

Parámetros de esta entidad:

Brillo
De 0 a 255.
Nivel de brillo del objeto que representa la vida.

Scale
Define el tamaño del objeto que representa la vida.

Transparencia
De 0 a 255.
Nivel de transparencia del objeto que representa la vida.
255 opaco, 0 completamente transparente, 128 semi-transparente.

Sombra
De 0 a 2.
Establece el tipo de sombra para este objeto.
Valor **0** sin sombra.
Valor **1** con sombra en círculo.
Valor **2** con sombra dinámica.
Si quieres ver ejemplos de estos tipos de sombras, consulta la sección de la entidad ConfigEnemigos.

---

*El objeto 3D que representa la vida es el fichero **'vida.act'** y está en la carpeta 'Actors'. Si quieres cambiarlo por otro, sólo tienes que copiar el nuevo fichero .act con ese mismo nombre en la misma carpeta.*
El nivel de brillo y transparencia para este objeto será el mismo en todo el nivel aunque variemos esos niveles en distintas entidades 'ItemVida'.

ItemHealth

Esta entidad representa 'kits' de salud que proporcionan al jugador un aumento en su nivel de salud.

Parámetros de esta entidad:

Brillo
De 0 a 255.
Nivel de brillo del objeto que representa el 'kit' de salud.

Scale
Define el tamaño del objeto que representa el 'kit' de salud.

Transparencia
De 0 a 255.
Nivel de transparencia del objeto que representa el 'kit' de salud.
255 opaco, 0 completamente transparente, 128 semi-transparente.

Num_ACT
De 1 a 3.
Establece que tipo de objeto 3D representa el 'kit' de salud.
Con 1 se utilizará el fichero 'medkit.act'
Con 2 se utilizará el fichero 'medkit2.act'
Con 3 se utilizará el fichero 'medkit3.act'

Incremento
De 1 a 100.

Cantidad de salud que recibirá el jugador al recoger este 'kit' de salud.

Reaparece

Con 'True' haremos que el 'kit' de salud reaparezca al cabo de unos segundos de haberlo recogido el jugador.

Sombra
De 0 a 2.
Establece el tipo de sombra para este objeto.
Valor **0** sin sombra.
Valor **1** con sombra en círculo.
Valor **2** con sombra dinámica.
Si quieres ver ejemplos de estos tipos de sombras, consulta la sección de la entidad ConfigEnemigos.

---

*Los objetos 3D que representan los 'kits' de salud están en la carpeta 'Actors'. Si quieres cambiarlos por otros, sólo tienes que copiar los nuevos ficheros .act con los nombres **'medkit.act', 'medkit2.act' o 'medkit3.act'** en la misma carpeta.*
Los niveles de brillo y transparencia de un tipo de 'kit' serán los mismos para el resto de 'kits' de ese tipo en todo el nivel, aunque variemos esos niveles en distintas entidades 'ItemHealth'.

ItemArmor

Al recoger este objeto, proporciona un aumento en el nivel de armadura del jugador.

Parámetros de esta entidad:

Brillo
De 0 a 255.
Nivel de brillo del objeto que representa la armadura.

Scale
Define el tamaño del objeto que representa la armadura.

Transparencia
De 0 a 255.
Nivel de transparencia del objeto que representa la armadura.
255 opaco, 0 completamente transparente, 128 semi-transparente.

Incremento
De 1 a 100.
Cantidad de armadura que recibirá el jugador al recoger este objeto.

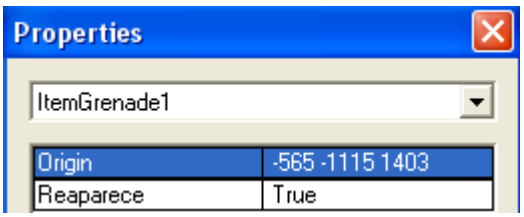
Reaparece
Con 'True' haremos que la armadura reaparezca al cabo de unos segundos de haberla recogido el jugador.

Sombra
De 0 a 2.
Establece el tipo de sombra para este objeto.
Valor **0** sin sombra.
Valor **1** con sombra en círculo.
Valor **2** con sombra dinámica.
Si quieres ver ejemplos de estos tipos de sombras, consulta la sección de la entidad ConfigEnemigos.

---

*El objeto 3D que representa la armadura es el fichero **'armor.act'** y está en la carpeta 'Actors'. Si quieres cambiarlo por otro, sólo tienes que copiar el nuevo fichero .act con ese mismo nombre en la misma carpeta.*
El nivel de brillo y transparencia para este objeto será el mismo en todo el nivel aunque variemos esos niveles en distintas entidades 'ItemArmor'.

ItemGrenade



Al recoger el jugador este objeto, le proporcionará el arma número 2, también llamada Lanza-Granadas en este tutorial.

Parámetros de esta entidad:

Origin

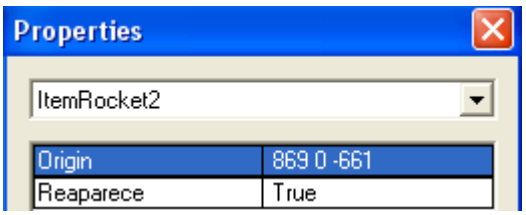
Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Reaparece

Con 'True' haremos que esta arma reaparezca al cabo de unos segundos de haber sido recogida por el jugador.

El objeto 3D que representa esta arma y la configuración de las diversas características de la misma, se realiza en la entidad 'ConfigArmaDos'.

ItemRocket



Al recoger el jugador este objeto, le proporcionará el arma número 3, también llamada Lanza-Cohetes en este tutorial.

Parámetros de esta entidad:

Origin

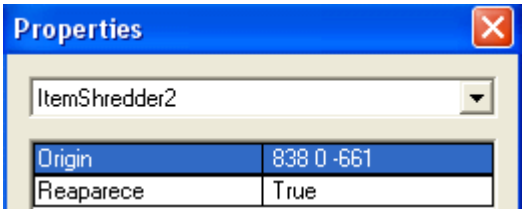
Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Reaparece

Con 'True' haremos que esta arma reaparezca al cabo de unos segundos de haber sido recogida por el jugador.

El objeto 3D que representa esta arma y la configuración de las diversas características de la misma, se realiza en la entidad 'ConfigArmaTres'.

ItemShredder



Al recoger el jugador este objeto, le proporcionará el arma número 4, también llamada Fusil o Shredder en este tutorial.

Parámetros de esta entidad:

Origin

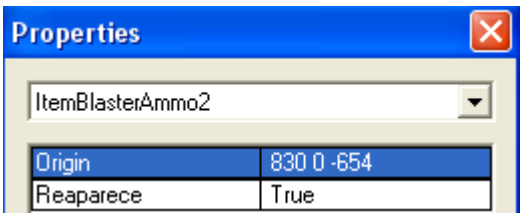
Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Reaparece

Con 'True' haremos que esta arma reaparezca al cabo de unos segundos de haber sido recogida por el jugador.

El objeto 3D que representa esta arma y la configuración de las diversas características de la misma, se realiza en la entidad 'ConfigArmaCuatro'.

ItemBlasterAmmo



Cargador de munición para el arma número 1.
Al recoger este objeto, proporciona un aumento en la munición del arma número 1 si la lleva el jugador. La cantidad de munición que aumentará la podemos variar en el parámetro 'Municion_cargador' de la entidad 'ConfigArmaUno'.

(con

Parámetros de esta entidad:

Origin

Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad el ratón) desde 'World Editor'.

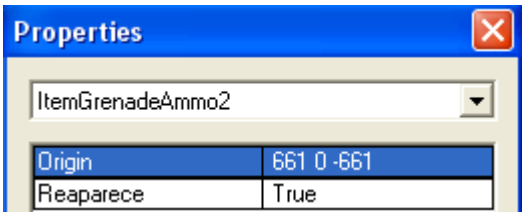
Reaparece

Con 'True' haremos que este cargador reaparezca al cabo de unos segundos de haberlo recogido el jugador.

---

*» El objeto 3D que representa este cargador es el fichero **'blaster_ammo.act'** y está en la carpeta **'Armas'**. Si quieres cambiarlo por otro, sólo tienes que copiar el nuevo fichero .act con ese mismo nombre en la misma carpeta.*
» No existe una entidad 'ItemBlaster' que sería el arma número 1 para recoger en el escenario, ya que esta arma es la única que siempre lleva el jugador. Recuerda que si quieres que esta arma nunca la lleve el jugador porque quieres hacer un juego sin armas o por lo que sea, puedes hacerlo mediante el parámetro 'Usar_Blaster' de la entidad 'DeathMathStart'.

ItemGrenadeAmmo



Granada que representa la munición para el arma número 2.
Al recoger este objeto, proporciona un aumento en la munición del arma número 2 si la lleva el jugador. La cantidad de munición que aumentará la podemos variar en el parámetro 'Municion_cargador' de la entidad 'ConfigArmaDos'.

Parámetros de esta entidad:

Origin

Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

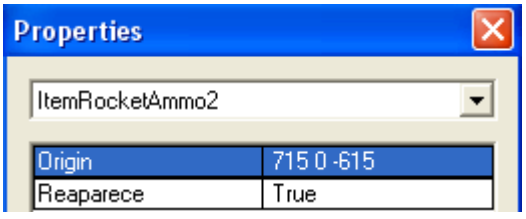
Reaparece

Con 'True' haremos que esta granada reaparezca al cabo de unos segundos de haberla recogido el jugador.

---

*El objeto 3D que representa a la granada es el fichero **'grenade.act'** y está en la carpeta **'Armas'**. Si quieres cambiarlo por otro, sólo tienes que copiar el nuevo fichero .act con ese mismo nombre en la misma carpeta.*

ItemRocketAmmo



Cargador de munición para el arma número 3.
Al recoger este objeto, proporciona un aumento en la munición del arma número 3 si la lleva el jugador. La cantidad de munición que aumentará la podemos variar en el parámetro 'Municion_cargador' de la entidad 'ConfigArmaTres.

Parámetros de esta entidad:

Origin

Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

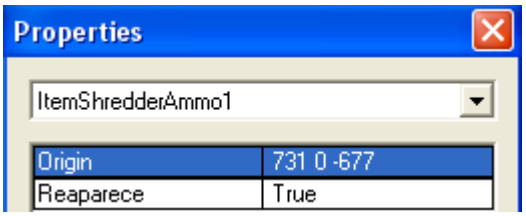
Reaparece

Con 'True' haremos que este cargador reaparezca al cabo de unos segundos de haberlo recogido el jugador.

---

*El objeto 3D que representa este cargador es el fichero **'rocket.act'** y está en la carpeta **'Armas'**. Si quieres cambiarlo por otro, sólo tienes que copiar el nuevo fichero .act con ese mismo nombre en la misma carpeta.*

ItemShredderAmmo



Cargador de munición para el arma número 4.
Al recoger este objeto, proporciona un aumento en la munición del arma número 4 si la lleva el jugador. La cantidad de munición que aumentará la podemos variar en el parámetro 'Municion_cargador' de la entidad 'ConfigArmaCuatro'.

Parámetros de esta entidad:

Origin

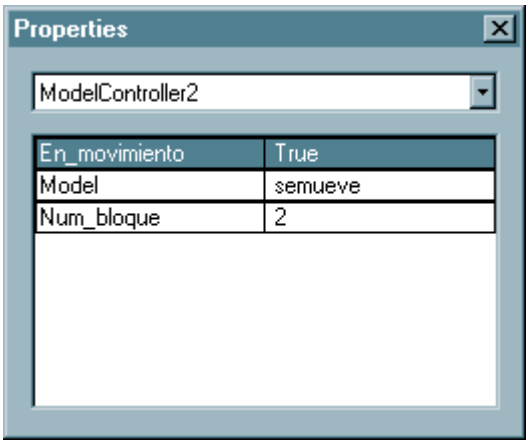
Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Reaparece

Con 'True' haremos que este cargador reaparezca al cabo de unos segundos de haberlo recogido el jugador.

*El objeto 3D que representa este cargador es el fichero **'sammo.act'** y está en la carpeta **'Armas'**. Si quieres cambiarlo por otro, sólo tienes que copiar el nuevo fichero .act con ese mismo nombre en la misma carpeta.*

ModelController



Esta entidad funciona de manera similar a la 'Door' o 'MovingPlat', se utiliza para asociarle un 'model' que hemos definido con una animación, como si lo fuéramos a asociar a una entidad 'Door', pero en lugar de eso lo asociamos a una entidad 'ModelController', de esa forma la animación se producirá en el escenario de manera continua y sin necesidad de que el jugador intervenga para activarla o desactivarla, aunque si lo deseamos podremos hacer que los bloques se muevan en función de lo que realice el jugador.

Parámetros de esta entidad:

Model

Aquí debemos seleccionar el 'model' que previamente hemos creado con los 'brushes' que formarán todo el bloque que deseamos animar.

Num_bloque

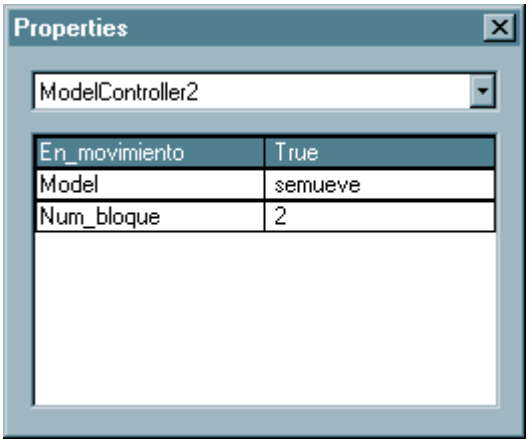
Número de bloque (entre 1 y 255).
Son números que deberemos asignar nosotros manualmente cada vez que creemos una nueva entidad 'ModelController'.
Este número será el que se utilizará en los comandos 'BLQ0' y 'BLQ1' para controlar posteriormente el movimiento del bloque.

En_movimiento

Con valor 'True' este bloque se moverá desde que de comienzo el nivel.
Con valor 'False' este bloque no comenzará a moverse al dar comienzo el nivel.

Recuerda que durante el transcurso del juego disponemos de los comandos 'BLQ1' y 'BLQ0' para activar y desactivar las animaciones de estos bloques, con lo que podremos parar y reanudar la animación de los bloques según el desarrollo de nuestro juego.
Como puedes ver, esta entidad nos permite dotar a nuestro nivel de zonas del escenario con animaciones de lo más variadas para dar más vida a nuestro mundo. Algunas de las utilidades de esta entidad podrían ser:

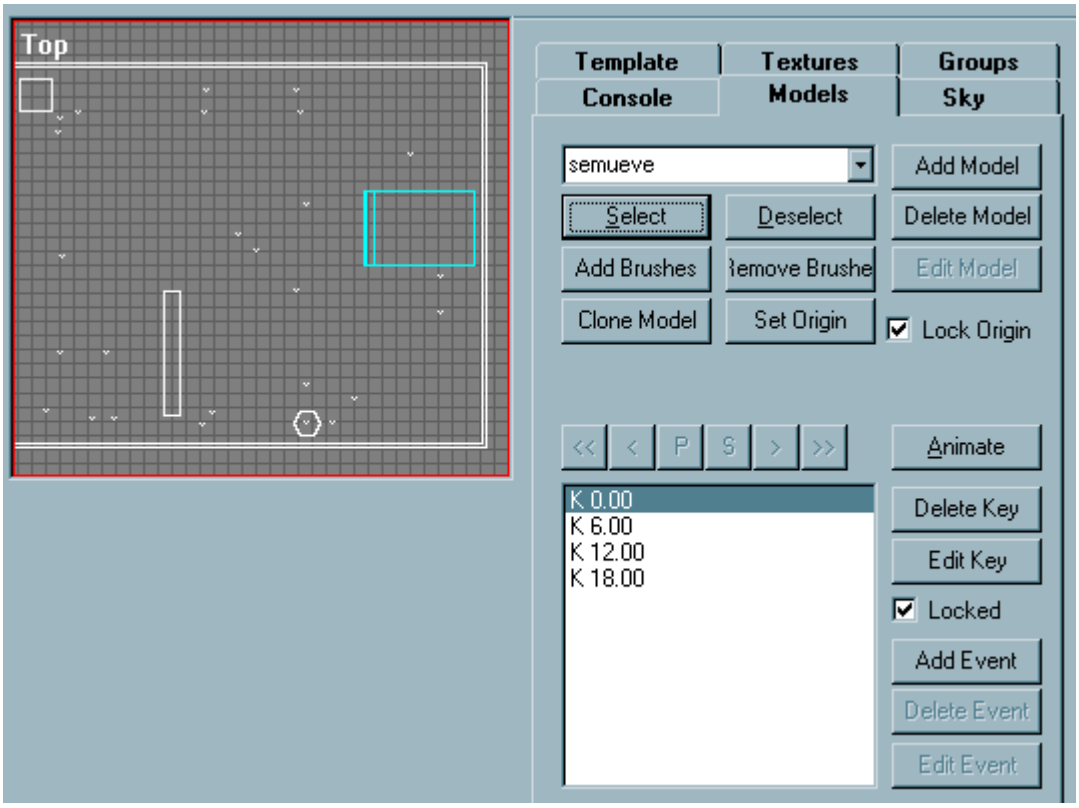
Plantear al jugador fases de habilidad donde tenga que actuar rápido y con inteligencia para sortear bloques que le arrastren a zonas con lava.



Utilizar un grupo de plataformas móviles para alcanzar un lugar.
Mover bloques tras los cuales se esconden objetos o enemigos.
Permitir al jugador acceder a otra zona del escenario.

Ojo con utilizar estos bloques móviles como ascensores o plataformas, observarás que si bien siguen haciendo la función de 'brushes' sólidos que bloquean el camino al jugador y a los enemigos, no transportarán ni al jugador ni a los enemigos si se suben a ellos.

Ejemplo: Una entidad 'ModelController' que controlará la animación de un 'model' llamado 'semueve':



Consulta las secciones de los comandos 'BLQ1' y 'BLQ0' para ver como activar y desactivar la animación de los bloques durante el transcurso del juego.

CamaraFija

Esta entidad nos permite colocar cámaras fijas (inmóviles) por el escenario, para poderlas activar durante el transcurso del juego y ofrecer al jugador otras vistas distintas del escenario, de zonas donde no se encuentre el jugador, o bien nuevas vistas del personaje protagonista que no sean las que proporcionan las cámaras de juego (primera persona, tercera persona o la cámara RPG).
Por defecto estas cámaras están simplemente colocadas en el escenario y no se activarán a menos que lo hagamos con el comando 'CAM1' o bien con el parámetro 'Activa' de la propia entidad 'CamaraFija'.

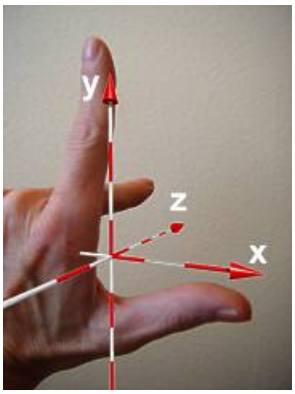
Parámetros de esta entidad:

Origin
Son tres cifras (X Y Z) que determinan la posición de esta entidad (en este caso de la cámara) en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Num_camara
Número de cámara (entre 1 y 64).
Son números que deberemos asignar nosotros manualmente cada vez que creemos una nueva entidad 'CamaraFija'.
Este número será el que se utilizará en los comandos 'CAM0' y 'CAM1' para desactivar y activar una cámara en concreto.

Activa
Con valor 'True' el nivel dará comienzo con la vista de esta cámara.
Con valor 'False' la cámara permanecerá inactiva al dar comienzo el nivel.

Angulo
Permite variar el ángulo hacia donde está apuntando la cámara. Las cámaras tienen tres ejes (X Y Z) para poder modificar su ángulo hacia cualquier dirección. Para girar un eje de la cámara, basta con introducir los grados de giro que queremos aplicar a ese eje.
Estos son los ejes X Y Z para estas cámaras:



Por ejemplo, tenemos una cámara con los ángulos X Y Z de esta manera: X=0 Y=0 Z=0. La vista sería esta:

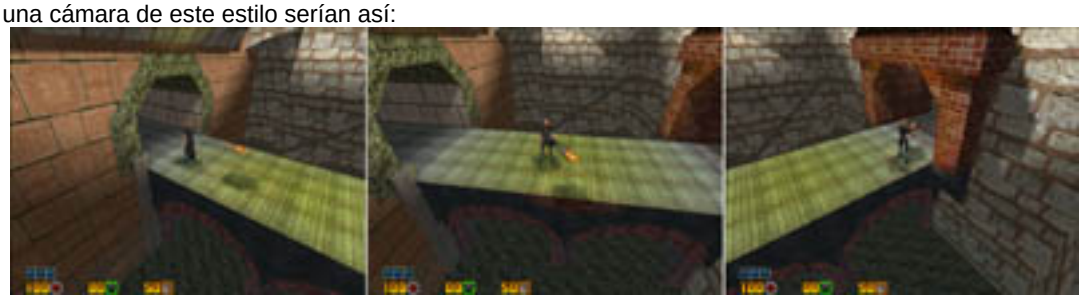
Para bajar esta vista 45 grados pondríamos X con -45 (negativo para bajar la cámara). La vista quedaría así:

Si hubiéramos puesto 45 de X, habríamos obtenido una vista del cielo:

... y esto es lo que obtenemos si ponemos Y con ángulo 30:

Apunta_al_jugador

Con valor 'True' la cámara apunta en todo momento al jugador.
Con valor 'False' la cámara permanecerá inmóvil, independientemente de los movimientos del jugador.
Con valor 'True' la cámara seguirá al personaje en sus movimientos por el escenario, eso no quiere decir que la cámara se desplace, simplemente enfocará continuamente y de forma automática al personaje protagonista que controla el jugador. Tres capturas de una cámara de este estilo serían así:



Con valor 'False' las capturas de la misma cámara tendrían este aspecto:

Tiempo

Tiempo (en segundos) que la cámara permanecerá activa una vez se haya activado con el comando 'CAM1' o bien si el juego ha comenzado con esta cámara activa. Transcurrido este tiempo esta cámara se desactivará automáticamente y el jugador pasará a tener la vista de la última cámara de juego que estaba utilizando.
Con tiempo 0 la cámara no se desactivará por tiempo.

Apuntar

Permite orientar la cámara hacia entidades 'Posicion' del escenario.
¡ Atención !, no se pueden usar los parámetros 'Apuntar_al_jugador' con 'True' y 'Apuntar' al mismo tiempo.

Model

Permite seleccionar un modelo asociado a la cámara.
La cámara quedará enganchada a ese modelo y le seguirá en todos sus movimientos.
Estos modelos son los bloques o conjunto de bloques definidos como 'model' que hayamos establecido en World Editor (pestaña 'models').
Está cámara si bien podrá rotar automáticamente para apuntar al jugador o a posiciones fijas (entidades 'Posicion'), no realizará las rotaciones del 'model' que tenga asociado.

Consulta las secciones de los comandos 'CAM1' y 'CAM0' para ver como activar y desactivar las cámaras durante el transcurso del juego.

Animacion

Esta entidad nos permite integrar animaciones (secuencias de imágenes .BMP) en nuestro juego en puntos concretos del escenario y que podremos activar mediante el comando 'ANI1' en el momento que queramos durante el transcurso del juego.

Parámetros de esta entidad:

Origin

Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Activa

Con valor 'True' esta animación se comenzará a mostrar desde que de comienzo el nivel.
Con valor 'False' esta animación no se comenzará a mostrar desde que de comienzo el nivel, habrá que activarla durante el transcurso del juego mediante un comando 'ANI1'.

Num_animacion

Número de animación (entre 1 y 255).
Son números que deberemos asignar nosotros manualmente.
Este número será el que se utilizará en el comando 'ANI1' para referirse a una animación en concreto.
Se puede utilizar el mismo número en diferentes entidades 'Animacion' si lo que queremos es presentar la misma animación en diferentes posiciones al mismo tiempo y utilizando un solo comando 'ANI1'.

Nombre

Nombre de la animación.
Con este nombre, Entidad 3D buscará en la carpeta 'bmp\texturas' una secuencia de imágenes .BMP que utilizará para presentar las animaciones.
Por ejemplo, si ponemos el nombre **chica** en el parámetro 'Nombre', Entidad 3D buscará los ficheros:
bmp\texturas\chica001.bmp
bmp\texturas\chica002.bmp
bmp\texturas\chica003.bmp
bmp\texturas\chica004.bmp
bmp\texturas\chica005.bmp
.
.
... hasta que encuentre el último.
También buscará los ficheros con el canal Alfa correspondiente a cada uno de los ficheros anteriores:
bmp\texturas\a_chica001.bmp
bmp\texturas\a_chica002.bmp
bmp\texturas\a_chica003.bmp
bmp\texturas\a_chica004.bmp
bmp\texturas\a_chica005.bmp
.
.
La función de estos ficheros con canal Alfa se explica más abajo en el apartado de ejemplos.

Cuadro_inicial

Número de cuadro, de la secuencia de ficheros .BMP, por el que se comenzará a presentar la animación.
Podemos tener una animación compuesta por 20 ficheros .BMP pero puede que nos interese mostrar en un momento dado sólo una secuencia que vaya por ejemplo desde el cuadro 10 al 15, entonces en el parámetro 'Cuadro_inicial' pondremos 10.

Cuadro_final

Número de cuadro, de la secuencia de ficheros .BMP, que será el último en la animación a presentar.
Podemos tener una animación compuesta por 20 ficheros .BMP pero puede que nos interese mostrar en un momento dado sólo una secuencia que vaya por ejemplo desde el cuadro 10 al 15, entonces en el parámetro 'Cuadro_final' pondremos 15.

Dimension

Tamaño que tendrá esta animación. A más valor mayor tamaño.

Velocidad

Velocidad de la animación. A menos valor, animación más lenta.

Sonido

Nombre del fichero de sonido (exclusivamente en formato .WAV) que se interpretará mientras se muestra la animación. No se debe indicar la extensión '.WAV'.
Este sonido es 3D posicional, por lo que el jugador oirá este sonido en el lado del altavoz que se encuentre el sonido en ese momento y con la atenuación correspondiente según la distancia en la que se encuentre el jugador.
Este fichero de sonido ha de estar en la carpeta 'wav' de Entidad 3D.

Por defecto se utiliza el fichero 'animacion.wav' que en realidad sólo produce silencio.

Transparencia

Nivel de transparencia que tendrá la animación (de 0 a 255), siendo 0 completamente invisible, 255 no es transparente y 128 semitransparente.

Comentarios sobre esta entidad:

Puedes poner tantas entidades 'Animacion' como quieras, pero sólo se podrá mostrar una secuencia de animación al mismo tiempo. Eso no quiere decir que no puedas mostrar, al mismo tiempo, la misma animación pero repetida cinco veces, pero todas mostrarán la misma secuencia, en el mismo tamaño y con las mismas características.
Asígnales un número de animación 'Num_animacion' distinto a cada una si las vas a activar por separado.
La animación siempre es vertical y gira en función de los movimientos del jugador para que siempre la vea de frente. Esto tiene la ventaja de poder mostrar objetos o personajes 2D en un mundo 3D sin que se note demasiado que son planos, ya que siempre se mostrarán al jugador de frente.
Los ficheros utilizados en la animación pueden ser con color RGB (16 millones de colores), aunque finalmente la animación los presentará a 256 colores.
La resolución de los ficheros .BMP ha de ser de: 64x64 o 128x128 o 256x256.
El nombre de la animación (parámetro 'Nombre') ha de ser igual en todas las entidades 'Animacion' de un nivel, si hay distintos nombres sólo se utilizará el que se encuentre en la primera entidad 'Animacion'. Para presentar distintas animaciones en un mismo nivel debes utilizar los parámetros 'Cuadro_inicial' y 'Cuadro_final' para presentar el intervalo de animación que desees en cada momento.
No se puede activar una animación mientras aún está corriendo otra animación, primero hay que detener la animación en curso con el comando 'ANI0'.
El total de ficheros .BMP que componen la animación no debe ser superior a 255.

¿ Para que sirve ?:

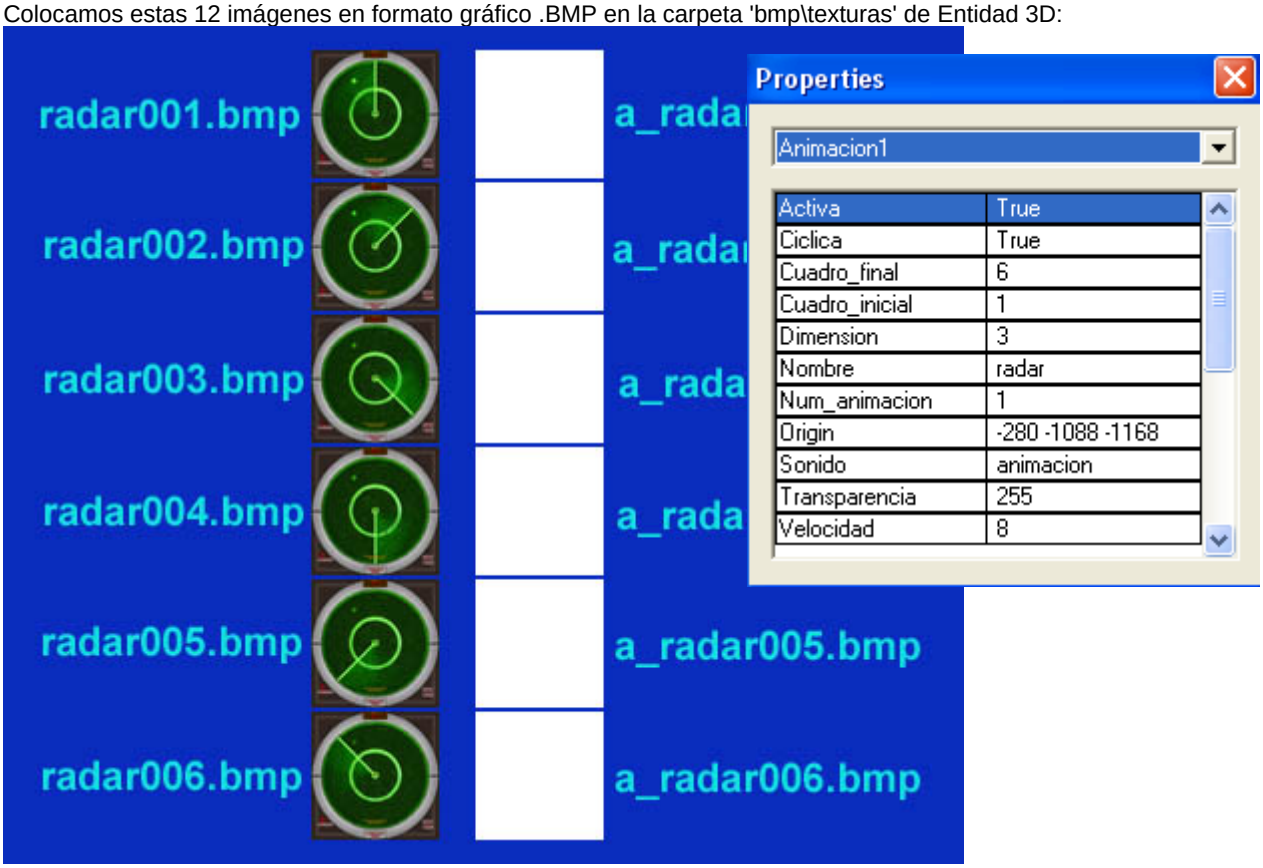
Esta entidad se puede utilizar para por ejemplo:

- Presentar animaciones a modo de pantallas que tenga que activar el jugador.
- Letreros luminosos de neón con letras parpadeantes (con sonido eléctrico incluido).
- Presentar personajes translúcidos que miran al jugador y le hablan.
- Anuncios publicitarios animados semitransparentes a modo de hologramas.
- Explosiones configurables (normales, de plasma, de gas, luminosas).
- Efectos especiales (espirales en movimiento, luces pulsantes).
- Imágenes fijas semitransparentes.

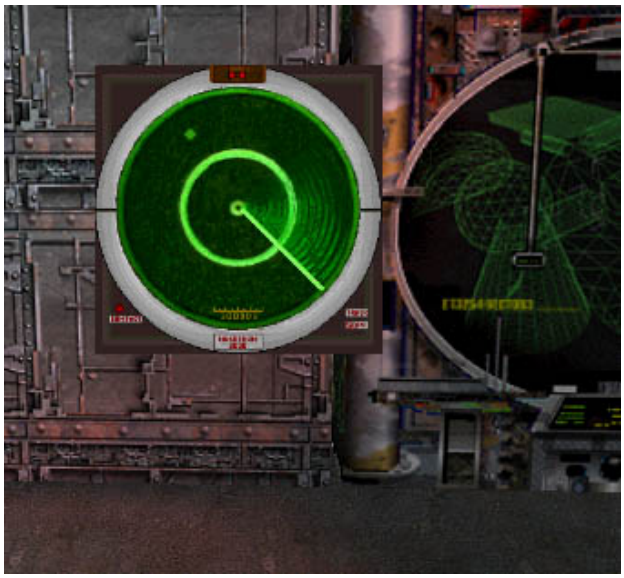
... todo dependerá de la calidad de los .BMP que utilices y de las variaciones que realices en los parámetros de la entidad 'Animacion' correspondiente.

Ejemplo 1:

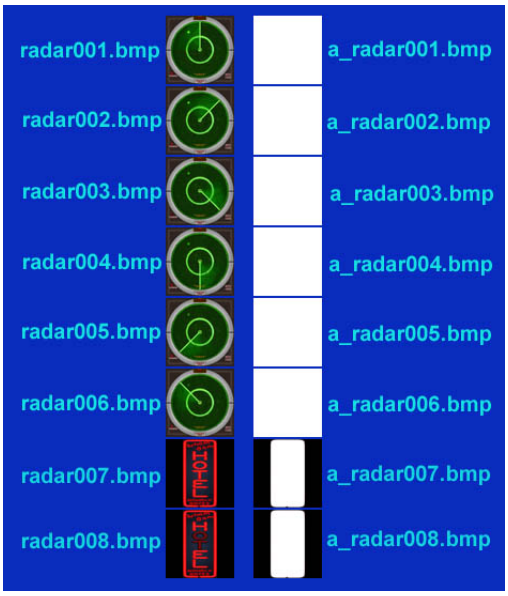
Queremos presentar una pantalla que muestre la animación de un radar barriendo la zona.
Colocamos estas 12 imágenes en formato gráfico .BMP en la carpeta 'bmp\texturas' de Entidad 3D:



6 ficheros que forman la animación y sus 6 ficheros con el canal Alfa (todos de un tamaño de 256x256).
¿ Para que sirven lo ficheros Alfa ?. Pues indican a Entidad 3D que zonas de cada uno de los gráficos será transparente y cuales no, siendo el blanco para indicar completamente opaco y negro para indicar completamente transparente. Como en el caso de nuestro radar no nos interesa hacer nada transparente, todos los cuadros tienen su fichero Alfa completamente relleno de blanco.
Para que aparezca esta animación en nuestro nivel sólo tendríamos que poner una entidad 'Animacion' con estos valores:



Ejemplo 2:



Queremos disponer de distintas animaciones en un mismo nivel para activar cada una de ellas en función de acciones del jugador.
Esta es la secuencia que necesitaríamos:

En esta ocasión hemos creado unos ficheros Alfa especiales para el letrero del Hotel para que en la animación sólo aparezca el letrero rojo sin las bandas negras que tienen a los lados los ficheros .BMP que contienen el letrero.
Para situar el letrero en nuestro nivel crearíamos una entidad 'Animacion' como esta:

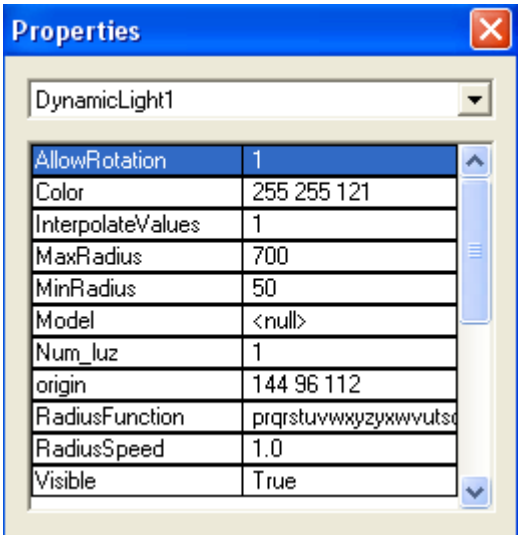
De esta forma tendríamos las dos animaciones cargadas en memoria, con sus características definidas con sus entidades correspondientes y situadas en el nivel en las posiciones donde se presentarán las animaciones.
Como sólo se puede mostrar una animación distinta al mismo tiempo, tendremos que utilizar los comandos 'ANI0 y ANI1' para desactivar y activar las animaciones durante el transcurso del juego, que en nuestro caso sería **'ACCI ANI1 1'** para activar el radar y **'ACCI ANI1 2'** para activar el letrero luminoso. Para desactivar cualquiera de las dos utilizaremos **'ACCI ANI0'**.

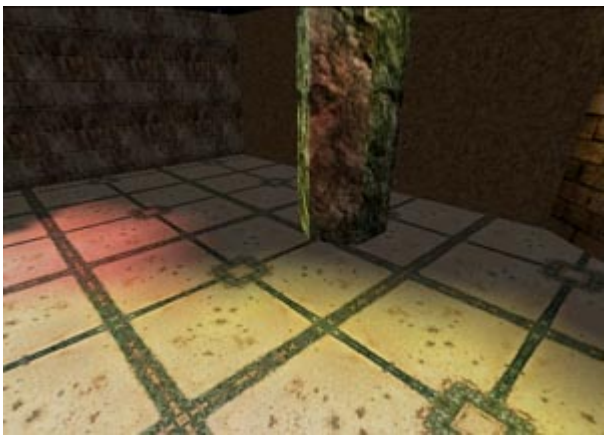
Efectos avanzados con los ficheros Alfa (Alpha Blending):

Los ficheros Alfa no sólo soportan los colores blanco y negro para indicar opacidad y transparencia respectivamente, sino que también se pueden utilizar escalas de grises para hacer semitransparencias en función del nivel de brillo del gris que utilicemos. En resumen, cuanto más clara sea la zona que definimos en el fichero Alfa más opaca se presentará la animación y cuanto más oscura más transparente.

Aquí puedes ver el resultado de utilizar estos matices en un fichero Alfa con grises:

DynamicLight





Se utiliza para situar luces en el escenario.

Parámetros de esta entidad:

Origin

Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Num_luz

Número de luz (entre 1 y 255).
Son números que deberemos asignar nosotros manualmente.
Este número será el que se utilizará en los comandos 'LUZ1' y 'LUZ0' para referirse a una luz en concreto.
Se puede utilizar el mismo número en diferentes entidades 'DynamicLight' si queremos manejar diferentes luces al mismo tiempo utilizando un solo comando 'LUZ1' o 'LUZ0'.

Model

Permite seleccionar un modelo asociado a la luz.
La luz quedará enganchada a ese modelo y le seguirá en todos sus movimientos.
Estos modelos son los bloques o conjunto de bloques definidos como 'model' que hayamos establecido en World Editor (pestaña 'models').

AllowRotation

Si la luz se asocia a un modelo, seguirá su movimiento de rotación si ponemos '1'.

InterpolateValues

Se utiliza para producir suaves variaciones en la luz (parpadeos). Si ponemos '0' las variaciones serán muy bruscas.

MaxRadius

El máximo radio de acción de la luz.

MinRadius

El mínimo radio de acción de la luz.

RadiusFunction

Aquí entraremos una secuencia de letras que determinarán los niveles de variación de la luz. La letra 'a' es para muy oscuro, mientras que la letra 'z' es para muy brillante.

RadiusSpeed

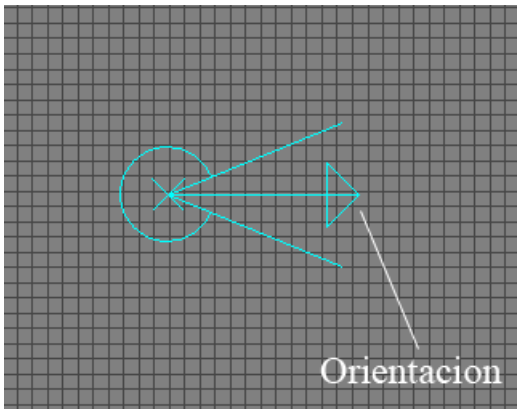
Velocidad de las variaciones de luz determinadas por 'RadiusFunction'.

Visible

Con 'True' la luz será visible desde el comienzo del nivel.
Con 'False' la luz no será visible al dar comienzo el nivel.

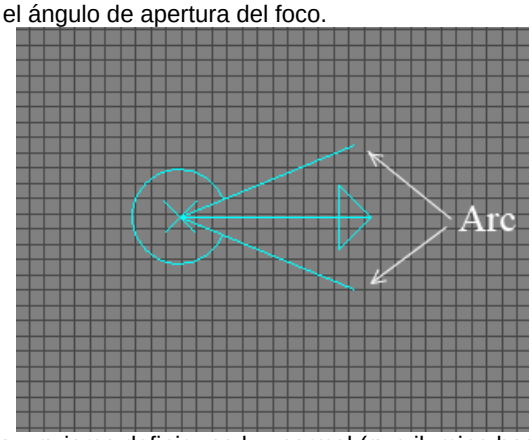
Orientacion

Junto con el parámetro '**Arc**' permite definir una luz dinámica tipo foco, que sólo iluminará hacia un punto en concreto.
Los parámetro X Y Z de este parámetro nos permiten rotar los ejes X Y Z para orientar el foco hacia donde queramos.

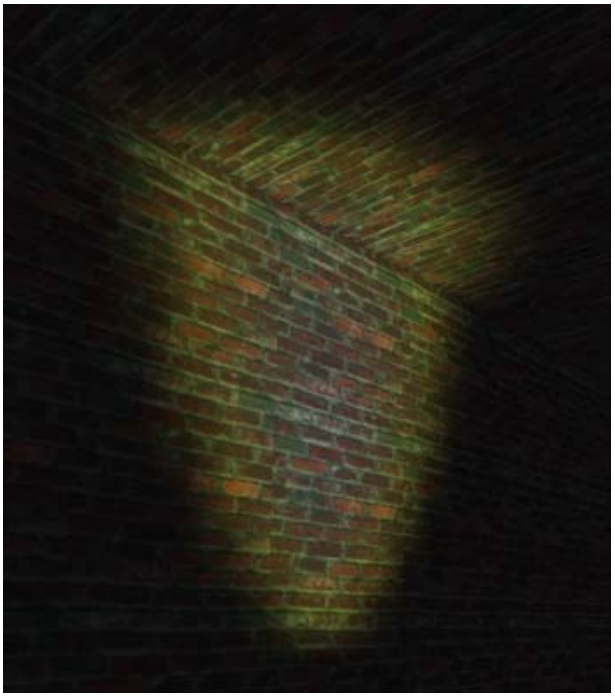


Arc

Junto con el parámetro '**Orientacion**' permite definir una luz dinámica tipo foco, que sólo iluminará hacia un punto en concreto.
El parámetro '**Arc**' permite establecer el ángulo de apertura del foco.



Si no quieres que sea una luz tipo foco y quieres definir una luz normal (que ilumine hacia todos los lados), deja con el valor 0 el parámetro 'Arc'.

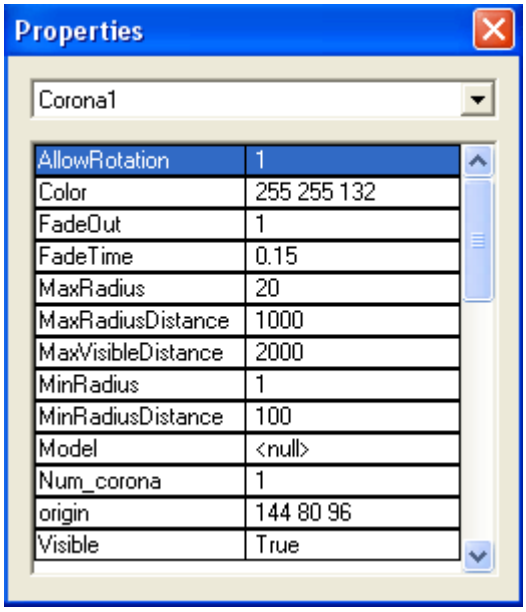


Luz tipo foco

Esta entidad, a diferencia de la entidad 'Light', puede producir variaciones de intensidad de luz durante el transcurso del juego, además de poderse desplazar por el escenario.
Asigna un número de luz 'Num_luz' distinto a cada una de la entidades 'DynamicLight' si las vas a activar por separado.

Consulta el manual de 'World Editor' para ver ejemplos de entidades 'DynamicLight'.
Consulta la sección de los comandos 'LUZ1' y 'LUZ0' para saber como activar y desactivar luces durante el transcurso del juego

Corona



Se utiliza para producir el efecto de resplandor en luces.

Parámetros de esta entidad:

Origin

Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Num_corona

Número de corona (entre 1 y 255).
Son números que deberemos asignar nosotros manualmente.
Este número será el que se utilizará en los comandos 'COR1' y 'COR0' para referirse a una corona en concreto.
Se puede utilizar el mismo número en diferentes entidades 'Corona' si queremos manejar diferentes coronas al mismo tiempo utilizando un solo comando 'COR1' o 'COR0'.

Model

Modelo asociado a la corona.
La corona quedará enganchada a ese modelo y le seguirá en todos sus movimientos.

AllowRotation

Si la corona se asocia a un modelo, seguirá su movimiento de rotación si ponemos '1'.

FadeOut

Nivel de desvanecimiento de la luz.

FadeTime

Tiempo en segundos para desvanecerse.

MaxRadius

Máximo radio de la corona.

MinRadius

Mínimo radio de la corona.

MaxRadiusDistance

Por encima de esta distancia, la corona se verá al tamaño 'MaxRadius'.

MinRadiusDistance

Por debajo de esta distancia, la corona se verá al tamaño 'MinRadius'.

MaxVisibleDistance

Distancia máxima a la que es visible la corona.

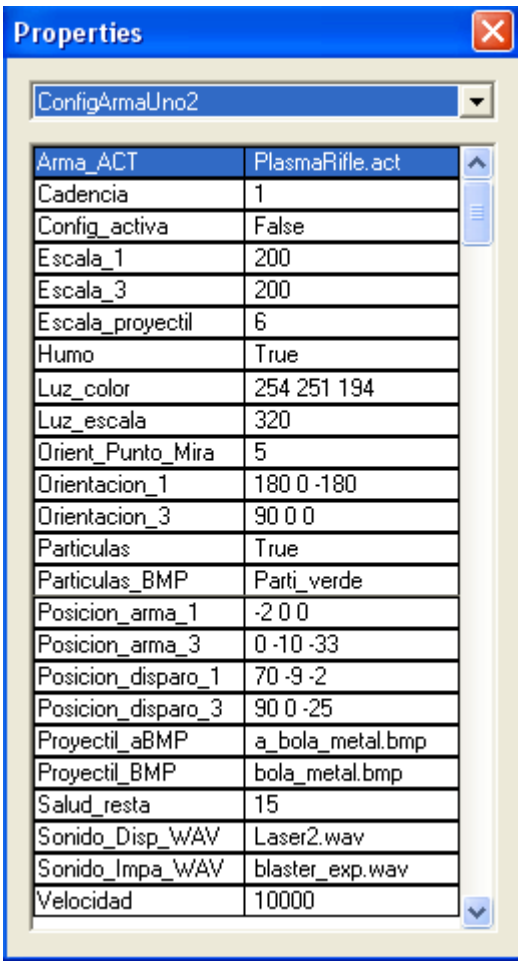
Visible

Con 'True' la corona será visible desde el comienzo del nivel.
Con 'False' la corona no será visible al dar comienzo el nivel.

Esta entidad no desprende luz que afecte al escenario ni a ningún elemento del mismo, sólo produce un efecto de resplandor en forma de estrella.
Para dar un poco más de realismo a las entidades de luz 'Light', 'DynamicLight' y 'Spotlight', podemos acompañarlas con entidades 'Corona' para producir el resplandor de esas fuentes de luz.
Asigna un número de corona 'Num_corona' distinto a cada una de la entidades 'Corona' si las vas a activar por separado.

Consulta el manual de 'World Editor' para ver ejemplos de entidades 'Corona'.
Consulta la sección de los comandos 'COR1' y 'COR0' para saber como activar y desactivar coronas durante el transcurso del juego.

ConfigArmaUno



Permite establecer una serie de parámetros para configurar el arma número uno (la que se selecciona pulsando la tecla '1'), permitiendo de esta forma modificar esta arma o crear una completamente nueva.

Parámetros de esta entidad:

Config_activa

Con valor 'True' esta configuración tomará efecto al dar comienzo el nivel.
Con valor 'False' esta configuración no tomará efecto al dar comienzo el nivel.
En un mismo nivel podemos tener más de una entidad 'ConfigArmaUno' con parámetros distintos para crear diferentes armas o efectos. Con 'Config_activa' estableceremos cual de esas entidades es la que se tomará como válida al comenzar el nivel, ya que sólo se permite una configuración activa al mismo tiempo, en el resto de entidades 'ConfigArmaUno' deberás poner 'False' para este parámetro.

Arma_ACT

Nombre del fichero .ACT que representa el arma que queremos ver.
Este fichero tiene que estar en la carpeta **'armas'** de Entidad 3D.

Sonido_Disp_WAV

Nombre del fichero de sonido .WAV que se reproducirá al disparar el arma.
Este fichero tiene que estar en la carpeta **'wav/armas'** de Entidad 3D.

Sonido_Impa_WAV

Nombre del fichero de sonido .WAV que se reproducirá cuando el proyectil impacte con algo o alguien.
Este fichero tiene que estar en la carpeta **'wav/armas'** de Entidad 3D.

Proyectil_BMP

Nombre del fichero .BMP que representa el proyectil que disparará el arma.
Este fichero tiene que estar en la carpeta **'bmp/weapon'** de Entidad 3D.
Las resoluciones permitidas son las habituales (32x32, 64x64, 128x128 y 256x256). La profundidad de color puede ser RGB (millones de colores) o 256 colores (color indexado).

Proyectil_aBMP

Nombre del fichero .BMP (con canal Alpha) que representa el proyectil.
Las mismas características que el parámetro 'Proyectil_BMP' pero con blancos y grises para establecer los niveles de transparencia del proyectil.
Consulta la sección 'Animacion' para saber que es un fichero con canal Alpha.

Escala_proyectil

Permite establecer el tamaño del proyectil que hemos definido en los parámetros 'Proyectil_BMP' y 'Proyectil_aBMP'.

Particulas_BMP

Nombre de la animación de partículas que seguirá al proyectil.
Con este nombre, Entidad 3D buscará en la carpeta **'bmp\fx'** una secuencia de 8 imágenes .BMP que utilizará para presentar una animación en toda la trayectoria del proyectil.
Por ejemplo, si ponemos el nombre **parti** en este parámetro, Entidad 3D buscará los ficheros:
bmp\fx\parti1.bmp

bmp\fx\parti2.bmp
bmp\fx\parti3.bmp
bmp\fx\parti4.bmp
bmp\fx\parti5.bmp
bmp\fx\parti6.bmp
bmp\fx\parti7.bmp
bmp\fx\parti8.bmp
Siempre han de ser 8 ficheros, ni uno más ni uno menos, si no dispones de al menos 8 ficheros para tu efecto, copia algunos de los BMPs que ya tengas con otros nombres hasta llegar a tener los ocho ficheros.
Puedes poner en la carpeta 'bmp\fx' tus propios efectos que hayas podido crear en ocho ficheros .BMP para conseguir personalizar tu propia arma. Estas animaciones pueden ser cualquier cosa, no necesariamente partículas: pueden ser espirales, difuminados, círculos animados, variaciones de colores, rayos, etc...
Las resoluciones permitidas para estos ficheros .BMP son las habituales (32x32, 64x64, 128x128 y 256x256). La profundidad de color puede ser RGB (millones de colores) o 256 colores (color indexado).
Este tipo de animación no necesita ficheros con canal Alpha, para este caso las zonas que se tomarán como transparentes serán las zonas oscuras de los propios ocho ficheros, por lo que cuanto más oscuro sea el color de una zona más transparente será, y cuanto más claro más opaco.

Particulas
Con valor 'True' se utilizará la animación establecida en el parámetro comentado anteriormente ('Particulas_BMP').
Con valor 'False' no se utilizará la animación establecida en el parámetro 'Particulas_BMP' ni ninguna otra animación.
Si no se utiliza la animación de partículas simplemente esta arma no presentará este efecto.

Velocidad
Velocidad del proyectil.
Controla la velocidad a la que se desplazará el proyectil una vez disparado. A mayor valor, mayor velocidad. Se permiten valores desde 600 a 50000.

Humo
Con valor 'True' se aplicará un efecto humo que seguirá al proyectil.
Con valor 'False' no se aplicará ningún efecto de humo al proyectil.

Orientacion_1
Estos tres valores de X Y y Z nos permitirán reorientar el arma (para un juego en primera persona) en caso de que no esté bien orientada. Esto lo podrás observar porque el arma te aparecerá girada apuntando hacia el jugador o hacia los lados, no hacia el frente como debería aparecer.
Estos valores se tomarán en grados, y se permiten desde 1 a 360 tanto negativos como positivos.
En prácticamente el 99% de las veces sólo tendrás que variar el valor Y y en saltos de 90 grados ya que como máximo el arma estará girada de la manera que se ha comentado anteriormente.
Si utilizas un arma que te aparece apuntando hacia arriba, hacia abajo o está ladeada, también tendrás que variar los valores X y Z.
El parámetro 'Orientacion_1' es de los primeros parámetros que debes establecer correctamente para seguir con el resto de parámetros de esta entidad a fin de conseguir colocar adecuadamente el arma.

Orientacion_3
Igual que el parámetro 'Orientacion_1' pero para vistas en tercera persona.
En esta vista en tercera persona intenta que el personaje empuñe el arma con la mano derecha cuando está disparando, y sitúala lo más horizontal posible en esa misma pose (parado y disparando). Del resto de animaciones y poses del personaje no tendrás que preocuparte ya que podrás observar que el personaje siempre mantendrá el arma cogida con la mano derecha de la misma forma que lo hayas establecido en la animación de disparar parado.

Posicion_arma_1
Con estos tres valores de X Y y Z podremos variar la colocación (para un juego en primera persona) del arma frente al jugador.
Variando X la moveremos adelante o atrás dependiendo del signo (+ o -) que utilicemos en el valor.
Variando Y la moveremos arriba o abajo dependiendo del signo (+ o -) que utilicemos en el valor.
Variando Z la moveremos a derecha o izquierda dependiendo del signo (+ o -) que utilicemos en el valor.
Esto es muy útil ya que las armas no son todas iguales ni del mismo tamaño y con la posibilidad de modificar su posición podremos situar el arma en la posición que más nos guste o que nos parezca más realista.
En la mayoría de las ocasiones, sólo bastará modificar el valor X para obtener una posición correcta si has establecido bien el parámetro 'Escala_1', por lo que te aconsejo que antes de ir variando sin control X Y y Z, juegues con los valores X y 'Escala_1' para situar correctamente el arma, en la inmensa mayoría de los casos será suficiente para colocar correctamente el arma frente al jugador.

Posicion_arma_3
Igual que el parámetro 'Posicion_arma_1' pero para vistas en tercera persona.

Orient_Punto_Mira
Orientación hacia el punto de mira en primera persona.
Permite acabar de corregir la orientación del arma para poder girarla hacia el punto de mira.
Este parámetro prácticamente no tendrás que tocarlo nunca si dejas el valor que hay por defecto.

Posicion_disparo_1
Con estos tres valores de X Y y Z podremos variar la posición de donde parte el proyectil disparado en juegos en primera persona.
Normalmente será justo delante del arma, pero puede haber armas en que la salida del proyectil no sea justo al frente.

Posicion_disparo_3
Igual que el parámetro 'Posicion_disparo_1' pero para vistas en tercera persona.

Escala_1
Nos permite establecer el tamaño con el que aparecerá el arma en primera persona. Este valor representa un porcentaje del tamaño original del arma.
Observarás que este valor puede variar bastante de una arma a otra ya que depende de la escala con la que se diseñó y creó el modelo 3D original del arma.

Escala_3
Igual que el parámetro 'Escala_1' pero para vistas en tercera persona.

Luz_color
Establece el color de una luz que acompañará al proyectil en todo su recorrido.
Si no quieres que aparezca esta luz, simplemente elige el color negro.

Luz_escala
Permite ajustar el tamaño de la luz que acompañará al proyectil.

Cadencia
Disparos por segundo para esta arma. Sólo se admiten valores del 1 al 10. A más valor, más disparos.

Salud_resta
Nivel de salud que se restará al jugador o enemigo que reciba un impacto directo del proyectil de esta arma.
De todos modos esta arma también producirá una reducción de salud si el jugador o enemigo se encuentran en el radio de acción de la onda expansiva en la zona de impacto del proyectil y que generalmente será 5 puntos menos de salud que lo que se descuenta por impacto directo.

Primera_persona
Se utiliza para indicar que el arma que se está utilizando es exclusivamente para la vista en primera persona.
Con `True' no se le permitirá al jugador cambiar a otras cámaras que no sea la cámara con vista en primera persona.
Con 'False' se le permitirá al jugador cambiar a las cámara que hayamos permitido mediante la entidad 'Configuracion'.
Poner 'True' es necesario si estamos utilizando un arma en la que se ven las manos o brazos del jugador que sostiene el arma, ya que este modelo 3D no nos serviría para una vista en tercera persona al contener precisamente esas manos o brazos cortados que sólo se ven bien en una vista en primera persona.

AnimDisparo
Nombre de la animación de disparo en primera persona que se reproducirá cuando el jugador dispare. Normalmente esta animación se llama 'Shoot'.
Para que la animación funcione, el fichero .ACT que utilices para esta arma debe contener el nombre de animación que quieres realizar.

AnimDispVel
Establece la velocidad a la que se reproducirá la animación definida en 'AnimDisparo'.
0.1=rápida 0.01=lenta 0.05=media

AnimNDisparo
Nombre de la animación del arma sin disparar en primera persona que se reproducirá mientras el jugador no dispare. Normalmente esta animación se llama 'Idle'.
Para que la animación funcione, el fichero .ACT que utilices para esta arma debe contener el nombre de animación que quieres realizar.

AnimNDispVel
Establece la velocidad a la que se reproducirá la animación definida en 'AnimNDisparo'.
0.1=rápida 0.01=lenta 0.05=media

Fogonazo_BMP
Nombre del fichero BMP que representa el fogonazo que se produce en la boca del cañón al disparar.
Este fichero tiene que estar en la carpeta **'bmp/explode'** de Entidad 3D.

Fogonazo_aBMP
Nombre del fichero BMP (con canal Alpha) que representa el fogonazo.
Este fichero tiene que estar en la carpeta **'bmp/explode'** de Entidad 3D.

Escala_fogonazo_1
Permite establecer el tamaño del fogonazo en primera persona.

Por defecto está a 0 pero si quieres verlo puedes comenzar con el valor 0.5.

Escala_fogonazo_3
Permite establecer el tamaño del fogonazo en tercera persona.
Por defecto está a 0 pero si quieres verlo puedes comenzar con el valor 1.5.

Municion_resta
Cantidad de munición a descontar en cada disparo.
Si pones 0 esta arma tendrá munición infinita.

Municion_inicial
Cantidad de munición inicial del arma número 1.
Se permiten valores de 1 a 100.
Esta cantidad será la que tendrá el jugador al dar comienzo el juego.

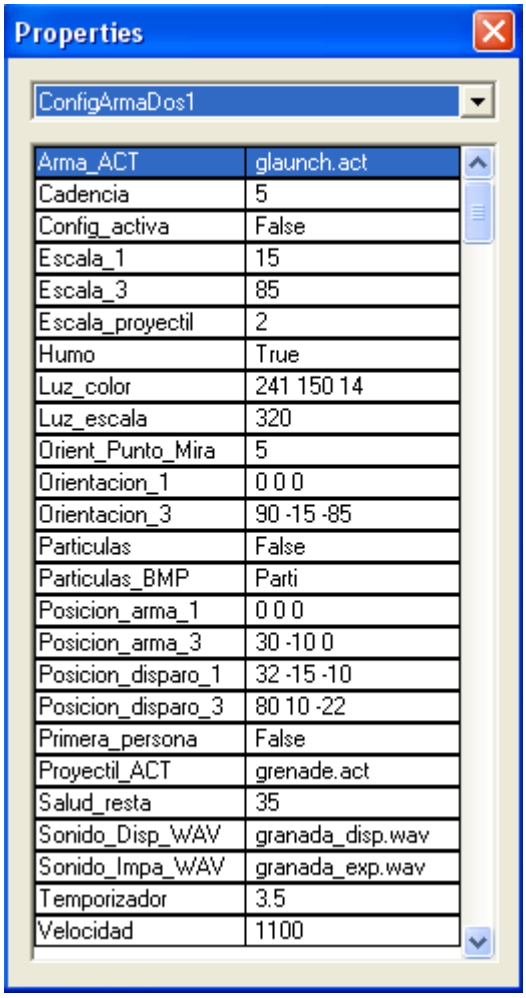
Municion_cargador
Cantidad de munición del cargador para esta arma.
Se permiten valores de 1 a 100.
Esta cantidad será la que se utilizará cuando el jugador recoja el cargador representado en el escenario con la entidad 'ItemBlasterAmmo'.

Resta_salud_critica
Cantidad de salud adicional que se restará al enemigo que reciba un impacto directo en su zona crítica, en el caso de que el enemigo tenga definida una zona crítica.
Esta cantidad de salud se sumará a la que ya se haya establecido en el parámetro 'Salud_resta'.
Consulta la entidad BotActorStart para saber como definir zonas críticas en los enemigos.

Comentarios sobre esta entidad:

Consulta la lista de armas disponibles para seleccionar el arma que te interese configurar.
Consulta la lista de proyectiles 2D disponibles para seleccionar el que te interese establecer.
Consulta la lista de partículas disponibles para seleccionar el efecto que te interese establecer.
Consulta la lista de sonidos disponibles para establecer sonidos tanto de disparo como de impacto para el arma.
Puedes utilizar una entidad 'ConfigArmaUno' con valores distintos en cada nivel de tu juego para que el jugador disponga de una arma distinta (como arma número uno) en cada nivel.
Esta entidad la puedes colocar donde quieras mientras se encuentre en el interior del mapa.
Piensa que todas las propiedades que definas para esta arma, serán también asignadas automáticamente para el mismo número de arma de los enemigos.
Las animaciones de las armas sólo funcionan en primera persona y para el jugador, si quieres realizar estos mismos efectos en un jugador en tercera persona o en enemigos, entonces la clave no es animar el arma sino el personaje, diseñando un personaje que incluya una animación 'Shoot' que realice el movimiento de brazos que te interese cuando esté parado y una animación 'Run' con la animación de correr del personaje y al mismo tiempo atacando.
En el caso de un enemigo, exactamente igual y además con el arma ya engancha en su mano derecha.
Recuerda que dispones del nivel de ejemplo 'tiro_al_pato' (distribuido junto a la versión 2.3a de Entidad 3D) que contiene numerosas entidades 'ConfigArmaUno' (una por cada arma) con cada arma correctamente situada y con un efecto ya pre-configurado.
Para activar cualquier arma en este nivel debes poner a 'True' el parámetro 'Config_activa' de la entidad 'ConfigArmaUno' que te interese activar como arma número 1, y poner 'False' en el mismo parámetro del resto de entidades 'ConfigArmaUno', finalmente tendrás que volver a recompilar el nivel.

ConfigArmaDos



Permite establecer una serie de parámetros para configurar el arma número dos (la que se selecciona pulsando la tecla '2'), permitiendo de esta forma modificar esta arma o crear una completamente nueva. Debido al comportamiento del proyectil, esta arma está pensada para ser utilizada para lanzar granadas, minas o cualquier otro objeto que pueda ser lanzado y que caerá al suelo a cierta distancia del jugador si no consigue impactar con alguien.

Parámetros de esta entidad:

Config_activa

Con valor 'True' esta configuración tomará efecto al dar comienzo el nivel.
Con valor 'False' esta configuración no tomará efecto al dar comienzo el nivel.
En un mismo nivel podemos tener más de una entidad 'ConfigArmaDos' con parámetros distintos para crear diferentes armas o efectos. Con 'Config_activa' estableceremos cual de esas entidades es la que se tomará como válida al comenzar el nivel, ya que sólo se permite una configuración activa al mismo tiempo, en el resto de entidades 'ConfigArmaDos' deberás poner 'False' para este parámetro.

Arma_ACT

Nombre del fichero .ACT que representa el arma que queremos ver.
Este fichero tiene que estar en la carpeta **'armas'** de Entidad 3D.

Sonido_Disp_WAV

Nombre del fichero de sonido .WAV que se reproducirá al disparar el arma.
Este fichero tiene que estar en la carpeta **'wav/armas'** de Entidad 3D.

Sonido_Impa_WAV

Nombre del fichero de sonido .WAV que se reproducirá cuando el proyectil impacte con algo o alguien.
Este fichero tiene que estar en la carpeta **'wav/armas'** de Entidad 3D.

Proyectil_ACT

Nombre del fichero .ACT que representa el proyectil que disparará el arma.
Este fichero tiene que estar en la carpeta **'armas'** de Entidad 3D.

Escala_proyectil

Permite establecer el tamaño del proyectil que hemos definido en los parámetros 'Proyectil_ACT'.

Particulas_BMP

Nombre de la animación de partículas que seguirá al proyectil.
Con este nombre, Entidad 3D buscará en la carpeta **'bmp\fx'** una secuencia de 8 imágenes .BMP que utilizará para presentar una animación en toda la trayectoria del proyectil.
Por ejemplo, si ponemos el nombre **parti** en este parámetro, Entidad 3D buscará los ficheros:
bmp\fx\parti1.bmp
bmp\fx\parti2.bmp
bmp\fx\parti3.bmp
bmp\fx\parti4.bmp
bmp\fx\parti5.bmp
bmp\fx\parti6.bmp
bmp\fx\parti7.bmp
bmp\fx\parti8.bmp

Siempre han de ser 8 ficheros, ni uno más ni uno menos, si no dispones de al menos 8 ficheros para tu efecto, copia algunos de los BMPs que ya tengas con otros nombres hasta llegar a tener los ocho ficheros.
Puedes poner en la carpeta 'bmp\fx' tus propios efectos que hayas podido crear en ocho ficheros .BMP para conseguir personalizar tu propia arma. Estas animaciones pueden ser cualquier cosa, no necesariamente partículas: pueden ser espirales, difuminados, círculos animados, variaciones de colores, rayos, etc...
Las resoluciones permitidas para estos ficheros .BMP son las habituales (32x32, 64x64, 128x128 y 256x256). La profundidad de color puede ser RGB (millones de colores) o 256 colores (color indexado).
Este tipo de animación no necesita ficheros con canal Alpha, para este caso las zonas que se tomarán como transparentes serán las zonas oscuras de los propios ocho ficheros, por lo que cuanto más oscuro sea el color de una zona más transparente será, y cuanto más claro más opaco.

Particulas
Con valor 'True' se utilizará la animación establecida en el parámetro comentado anteriormente ('Particulas_BMP').
Con valor 'False' no se utilizará la animación establecida en el parámetro 'Particulas_BMP' ni ninguna otra animación.
Si no se utiliza la animación de partículas simplemente esta arma no presentará este efecto.

Velocidad
Velocidad del proyectil.
Controla la velocidad a la que se desplazará el proyectil una vez disparado. A mayor valor, mayor velocidad. Se permiten valores desde 100 a 10000.

Humo
Con valor 'True' se aplicará un efecto humo que seguirá al proyectil.
Con valor 'False' no se aplicará ningún efecto de humo al proyectil.

Orientacion_1
Estos tres valores de X Y y Z nos permitirán reorientar el arma (para un juego en primera persona) en caso de que no esté bien orientada. Esto lo podrás observar porque el arma te aparecerá girada apuntando hacia el jugador o hacia los lados, no hacia el frente como debería aparecer.
Estos valores se tomarán en grados, y se permiten desde 1 a 360 tanto negativos como positivos.
En prácticamente el 99% de las veces sólo tendrás que variar el valor Y y en saltos de 90 grados ya que como máximo el arma estará girada de la manera que se ha comentado anteriormente.
Si utilizas un arma que te aparece apuntando hacia arriba, hacia abajo o está ladeada, también tendrás que variar los valores X y Z.
El parámetro 'Orientacion_1' es de los primeros parámetros que debes establecer correctamente para seguir con el resto de parámetros de esta entidad a fin de conseguir colocar adecuadamente el arma.

Orientacion_3
Igual que el parámetro 'Orientacion_1' pero para vistas en tercera persona.
En esta vista en tercera persona intenta que el personaje empuñe el arma con la mano derecha cuando está disparando, y sitúala lo más horizontal posible en esa misma pose (parado y disparando). Del resto de animaciones y poses del personaje no tendrás que preocuparte ya que podrás observar que el personaje siempre mantendrá el arma cogida con la mano derecha de la misma forma que lo hayas establecido en la animación de disparar parado.

Posicion_arma_1
Con estos tres valores de X Y y Z podremos variar la colocación (para un juego en primera persona) del arma frente al jugador.
Variando X la moveremos adelante o atrás dependiendo del signo (+ o -) que utilicemos en el valor.
Variando Y la moveremos arriba o abajo dependiendo del signo (+ o -) que utilicemos en el valor.
Variando Z la moveremos a derecha o izquierda dependiendo del signo (+ o -) que utilicemos en el valor.
Esto es muy útil ya que las armas no son todas iguales ni del mismo tamaño y con la posibilidad de modificar su posición podremos situar el arma en la posición que más nos guste o que nos parezca más realista.
En la mayoría de las ocasiones, sólo bastará modificar el valor X para obtener una posición correcta si has establecido bien el parámetro 'Escala_1', por lo que te aconsejo que antes de ir variando sin control X Y y Z, juegues con los valores X y 'Escala_1' para situar correctamente el arma, en la inmensa mayoría de los casos será suficiente para colocar correctamente el arma frente al jugador.

Posicion_arma_3
Igual que el parámetro 'Posicion_arma_1' pero para vistas en tercera persona.

Orient_Punto_Mira
Orientación hacia el punto de mira en primera persona.
Permite acabar de corregir la orientación del arma para poder girarla hacia el punto de mira.
Este parámetro prácticamente no tendrás que tocarlo nunca si dejas el valor que hay por defecto.

Posicion_disparo_1
Con estos tres valores de X Y y Z podremos variar la posición de donde parte el proyectil disparado en juegos en primera persona.
Normalmente será justo delante del arma, pero puede haber armas en que la salida del proyectil no sea justo al frente.

Posicion_disparo_3

Igual que el parámetro 'Posicion_disparo_1' pero para vistas en tercera persona.

Escala_1
Nos permite establecer el tamaño con el que aparecerá el arma en primera persona. Este valor representa un porcentaje del tamaño original del arma.
Observarás que este valor puede variar bastante de una arma a otra ya que depende de la escala con la que se diseñó y creó el modelo 3D original del arma.

Escala_3
Igual que el parámetro 'Escala_1' pero para vistas en tercera persona.

Luz_color
Establece el color de una luz que acompañará al proyectil en todo su recorrido.
Si no quieres que aparezca esta luz, simplemente elige el color negro.

Luz_escala
Permite ajustar el tamaño de la luz que acompañará al proyectil.

Cadencia
Disparos por segundo para esta arma. Sólo se admiten valores del 1 al 10. A más valor, más disparos.

Salud_resta
Nivel de salud que se restará al jugador o enemigo que reciba un impacto directo del proyectil de esta arma.
De todos modos esta arma también producirá una reducción de salud si el jugador o enemigo se encuentran en el radio de acción de la onda expansiva en la zona de impacto del proyectil y que generalmente será 5 puntos menos de salud que lo que se descuenta por impacto directo.

Primera_persona
Se utiliza para indicar que el arma que se está utilizando es exclusivamente para la vista en primera persona.
Con `True' no se le permitirá al jugador cambiar a otras cámaras que no sea la cámara con vista en primera persona.
Con 'False' se le permitirá al jugador cambiar a las cámara que hayamos permitido mediante la entidad 'Configuracion'.
Poner 'True' es necesario si estamos utilizando un arma en la que se ven las manos o brazos del jugador que sostiene el arma, ya que este modelo 3D no nos serviría para una vista en tercera persona al contener precisamente esas manos o brazos cortados que sólo se ven bien en una vista en primera persona.

Temporizador
Permite establecer un retardo de tiempo para la explosión del proyectil que hayamos lanzado.
Utiliza el valor 0 para que no explote por tiempo si no solamente por contacto, ideal para colocar minas.

AnimDisparo
Nombre de la animación de disparo en primera persona que se reproducirá cuando el jugador dispare. Normalmente esta animación se llama 'Shoot'.
Para que la animación funcione, el fichero .ACT que utilices para esta arma debe contener el nombre de animación que quieres realizar.

AnimDispVel
Establece la velocidad a la que se reproducirá la animación definida en 'AnimDisparo'.
0.1=rápida 0.01=lenta 0.05=media

AnimNDisparo
Nombre de la animación del arma sin disparar en primera persona que se reproducirá mientras el jugador no dispare.
Normalmente esta animación se llama 'Idle'.
Para que la animación funcione, el fichero .ACT que utilices para esta arma debe contener el nombre de animación que quieres realizar.

AnimNDispVel
Establece la velocidad a la que se reproducirá la animación definida en 'AnimNDisparo'.
0.1=rápida 0.01=lenta 0.05=media

Municion_resta
Cantidad de munición a descontar en cada disparo.
Si pones 0 esta arma tendrá munición infinita.

Municion_inicial
Cantidad de munición inicial del arma al recogerla por primera vez.
Se permiten valores de 1 a 100.
Esta cantidad será la que se entregará al jugador al recoger por primera vez esta arma, representada en el escenario con la entidad 'ItemGrenade'.

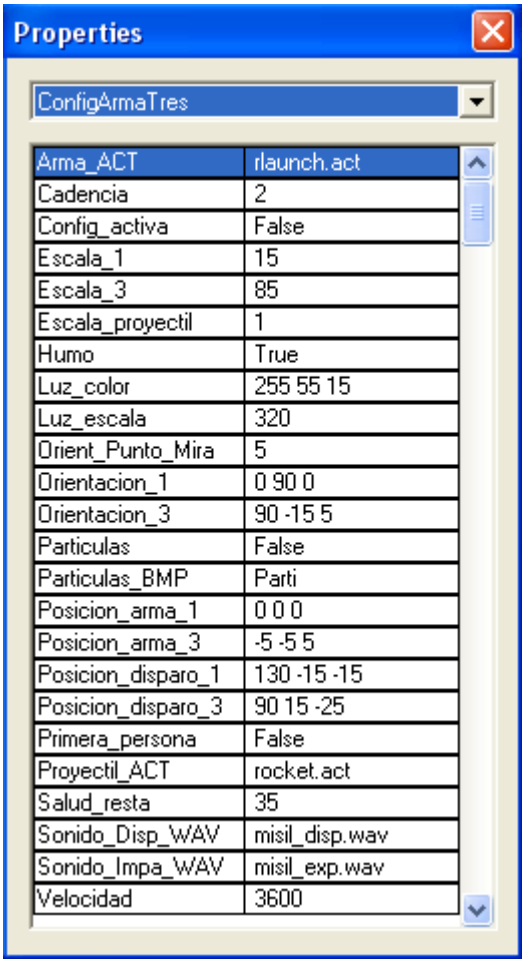
Municion_cargador
Cantidad de munición del cargador para esta arma.
Se permiten valores de 1 a 100.

Esta cantidad será la que se utilizará cuando el jugador recoja el cargador representado en el escenario con la entidad 'ItemGrenadeAmmo'.

Comentarios sobre esta entidad:

Consulta la lista de armas disponibles para seleccionar el arma que te interese configurar.
Consulta la lista de proyectiles 3D disponibles para seleccionar el que te interese establecer.
Consulta la lista de partículas disponibles para seleccionar el efecto que te interese establecer.
Consulta la lista de sonidos disponibles para establecer sonidos tanto de disparo como de impacto para el arma.
Puedes utilizar una entidad 'ConfigArmaDos' con valores distintos en cada nivel de tu juego para que el jugador disponga de una arma distinta (como arma número dos) en cada nivel.
Esta entidad la puedes colocar donde quieras mientras se encuentre en el interior del mapa.
Piensa que todas las propiedades que definas para esta arma, serán también asignadas automáticamente para el mismo número de arma de los enemigos.
Las animaciones de las armas sólo funcionan en primera persona y para el jugador, si quieres realizar estos mismos efectos en un jugador en tercera persona o en enemigos, entonces la clave no es animar el arma sino el personaje, diseñando un personaje que incluya una animación 'Shoot' que realice el movimiento de brazos que te interese cuando esté parado y una animación 'Run' con la animación de correr del personaje y al mismo tiempo atacando.
En el caso de un enemigo, exactamente igual y además con el arma ya engancha en su mano derecha.

ConfigArmaTres



Permite establecer una serie de parámetros para configurar el arma número tres (la que se selecciona pulsando la tecla '2'), permitiendo de esta forma modificar esta arma o crear una completamente nueva.
Debido al comportamiento del proyectil, esta arma está pensada para ser utilizada para lanzar algún tipo de cohete.

Parámetros de esta entidad:

Config_activa
Con valor 'True' esta configuración tomará efecto al dar comienzo el nivel.
Con valor 'False' esta configuración no tomará efecto al dar comienzo el nivel.
En un mismo nivel podemos tener más de una entidad 'ConfigArmaTres' con parámetros distintos para crear diferentes armas o efectos. Con 'Config_activa' estableceremos cual de esas entidades es la que se tomará como válida al comenzar el nivel, ya que sólo se permite una configuración activa al mismo tiempo, en el resto de entidades 'ConfigArmaTres' deberás poner 'False' para este parámetro.

Arma_ACT
Nombre del fichero .ACT que representa el arma que queremos ver.
Este fichero tiene que estar en la carpeta **'armas'** de Entidad 3D.

Sonido_Disp_WAV
Nombre del fichero de sonido .WAV que se reproducirá al disparar el arma.
Este fichero tiene que estar en la carpeta **'wav/armas'** de Entidad 3D.

Sonido_Impa_WAV
Nombre del fichero de sonido .WAV que se reproducirá cuando el proyectil impacte con algo o alguien.
Este fichero tiene que estar en la carpeta **'wav/armas'** de Entidad 3D.

Proyectil_ACT
Nombre del fichero .ACT que representa el proyectil que disparará el arma.

Este fichero tiene que estar en la carpeta **'armas'** de Entidad 3D.

Escala_proyectil
Permite establecer el tamaño del proyectil que hemos definido en los parámetros 'Proyectil_ACT'.

Particulas_BMP
Nombre de la animación de partículas que seguirá al proyectil.
Con este nombre, Entidad 3D buscará en la carpeta **'bmp\fx'** una secuencia de 8 imágenes .BMP que utilizará para presentar una animación en toda la trayectoria del proyectil.
Por ejemplo, si ponemos el nombre **parti** en este parámetro, Entidad 3D buscará los ficheros:
bmp\fx\parti1.bmp
bmp\fx\parti2.bmp
bmp\fx\parti3.bmp
bmp\fx\parti4.bmp
bmp\fx\parti5.bmp
bmp\fx\parti6.bmp
bmp\fx\parti7.bmp
bmp\fx\parti8.bmp
Siempre han de ser 8 ficheros, ni uno más ni uno menos, si no dispones de al menos 8 ficheros para tu efecto, copia algunos de los BMPs que ya tengas con otros nombres hasta llegar a tener los ocho ficheros.
Puedes poner en la carpeta 'bmp\fx' tus propios efectos que hayas podido crear en ocho ficheros .BMP para conseguir personalizar tu propia arma. Estas animaciones pueden ser cualquier cosa, no necesariamente partículas: pueden ser espirales, difuminados, círculos animados, variaciones de colores, rayos, etc...
Las resoluciones permitidas para estos ficheros .BMP son las habituales (32x32, 64x64, 128x128 y 256x256). La profundidad de color puede ser RGB (millones de colores) o 256 colores (color indexado).
Este tipo de animación no necesita ficheros con canal Alpha, para este caso las zonas que se tomarán como transparentes serán las zonas oscuras de los propios ocho ficheros, por lo que cuanto más oscuro sea el color de una zona más transparente será, y cuanto más claro más opaco.

Particulas
Con valor 'True' se utilizará la animación establecida en el parámetro comentado anteriormente ('Particulas_BMP').
Con valor 'False' no se utilizará la animación establecida en el parámetro 'Particulas_BMP' ni ninguna otra animación.
Si no se utiliza la animación de partículas simplemente esta arma no presentará este efecto.

Velocidad
Velocidad del proyectil.
Controla la velocidad a la que se desplazará el proyectil una vez disparado. A mayor valor, mayor velocidad. Se permiten valores desde 600 a 50000.

Humo
Con valor 'True' se aplicará un efecto humo que seguirá al proyectil.
Con valor 'False' no se aplicará ningún efecto de humo al proyectil.

Orientacion_1
Estos tres valores de X Y y Z nos permitirán reorientar el arma (para un juego en primera persona) en caso de que no esté bien orientada. Esto lo podrás observar porque el arma te aparecerá girada apuntando hacia el jugador o hacia los lados, no hacia el frente como debería aparecer.
Estos valores se tomarán en grados, y se permiten desde 1 a 360 tanto negativos como positivos.
En prácticamente el 99% de las veces sólo tendrás que variar el valor Y y en saltos de 90 grados ya que como máximo el arma estará girada de la manera que se ha comentado anteriormente.
Si utilizas un arma que te aparece apuntando hacia arriba, hacia abajo o está ladeada, también tendrás que variar los valores X y Z.
El parámetro 'Orientacion_1' es de los primeros parámetros que debes establecer correctamente para seguir con el resto de parámetros de esta entidad a fin de conseguir colocar adecuadamente el arma.

Orientacion_3
Igual que el parámetro 'Orientacion_1' pero para vistas en tercera persona.
En esta vista en tercera persona intenta que el personaje empuñe el arma con la mano derecha cuando está disparando, y sitúala lo más horizontal posible en esa misma pose (parado y disparando). Del resto de animaciones y poses del personaje no tendrás que preocuparte ya que podrás observar que el personaje siempre mantendrá el arma cogida con la mano derecha de la misma forma que lo hayas establecido en la animación de disparar parado.

Posicion_arma_1
Con estos tres valores de X Y y Z podremos variar la colocación (para un juego en primera persona) del arma frente al jugador.
Variando X la moveremos adelante o atrás dependiendo del signo (+ o -) que utilicemos en el valor.
Variando Y la moveremos arriba o abajo dependiendo del signo (+ o -) que utilicemos en el valor.
Variando Z la moveremos a derecha o izquierda dependiendo del signo (+ o -) que utilicemos en el valor.
Esto es muy útil ya que las armas no son todas iguales ni del mismo tamaño y con la posibilidad de modificar su posición podremos situar el arma en la posición que más nos guste o que nos parezca más realista.
En la mayoría de las ocasiones, sólo bastará modificar el valor X para obtener una posición correcta si has establecido bien el parámetro 'Escala_1', por lo que te aconsejo que antes de ir variando sin control X Y y Z, juegues con los valores X y

'Escala_1' para situar correctamente el arma, en la inmensa mayoría de los casos será suficiente para colocar correctamente el arma frente al jugador.

Posicion_arma_3
Igual que el parámetro 'Posicion_arma_1' pero para vistas en tercera persona.

Orient_Punto_Mira
Orientación hacia el punto de mira en primera persona.
Permite acabar de corregir la orientación del arma para poder girarla hacia el punto de mira.
Este parámetro prácticamente no tendrás que tocarlo nunca si dejas el valor que hay por defecto.

Posicion_disparo_1
Con estos tres valores de X Y y Z podremos variar la posición de donde parte el proyectil disparado en juegos en primera persona.
Normalmente será justo delante del arma, pero puede haber armas en que la salida del proyectil no sea justo al frente.

Posicion_disparo_3
Igual que el parámetro 'Posicion_disparo_1' pero para vistas en tercera persona.

Escala_1
Nos permite establecer el tamaño con el que aparecerá el arma en primera persona. Este valor representa un porcentaje del tamaño original del arma.
Observarás que este valor puede variar bastante de una arma a otra ya que depende de la escala con la que se diseñó y creó el modelo 3D original del arma.

Escala_3
Igual que el parámetro 'Escala_1' pero para vistas en tercera persona.

Luz_color
Establece el color de una luz que acompañará al proyectil en todo su recorrido.
Si no quieres que aparezca esta luz, simplemente elige el color negro.

Luz_escala
Permite ajustar el tamaño de la luz que acompañará al proyectil.

Cadencia
Disparos por segundo para esta arma. Sólo se admiten valores del 1 al 10. A más valor, más disparos.

Salud_resta
Nivel de salud que se restará al jugador o enemigo que reciba un impacto directo del proyectil de esta arma.
De todos modos esta arma también producirá una reducción de salud si el jugador o enemigo se encuentran en el radio de acción de la onda expansiva en la zona de impacto del proyectil y que generalmente será 5 puntos menos de salud que lo que se descuenta por impacto directo.

Primera_persona
Se utiliza para indicar que el arma que se está utilizando es exclusivamente para la vista en primera persona.
Con `True' no se le permitirá al jugador cambiar a otras cámaras que no sea la cámara con vista en primera persona.
Con 'False' se le permitirá al jugador cambiar a las cámara que hayamos permitido mediante la entidad 'Configuracion'.
Poner 'True' es necesario si estamos utilizando un arma en la que se ven las manos o brazos del jugador que sostiene el arma, ya que este modelo 3D no nos serviría para una vista en tercera persona al contener precisamente esas manos o brazos cortados que sólo se ven bien en una vista en primera persona.

AnimDisparo
Nombre de la animación de disparo en primera persona que se reproducirá cuando el jugador dispare. Normalmente esta animación se llama 'Shoot'.
Para que la animación funcione, el fichero .ACT que utilices para esta arma debe contener el nombre de animación que quieres realizar.

AnimDispVel
Establece la velocidad a la que se reproducirá la animación definida en 'AnimDisparo'.
0.1=rápida 0.01=lenta 0.05=media

AnimNDisparo
Nombre de la animación del arma sin disparar en primera persona que se reproducirá mientras el jugador no dispare. Normalmente esta animación se llama 'Idle'.
Para que la animación funcione, el fichero .ACT que utilices para esta arma debe contener el nombre de animación que quieres realizar.

AnimNDispVel
Establece la velocidad a la que se reproducirá la animación definida en 'AnimNDisparo'.
0.1=rápida 0.01=lenta 0.05=media

Municion_resta
Cantidad de munición a descontar en cada disparo.
Si pones 0 esta arma tendrá munición infinita.

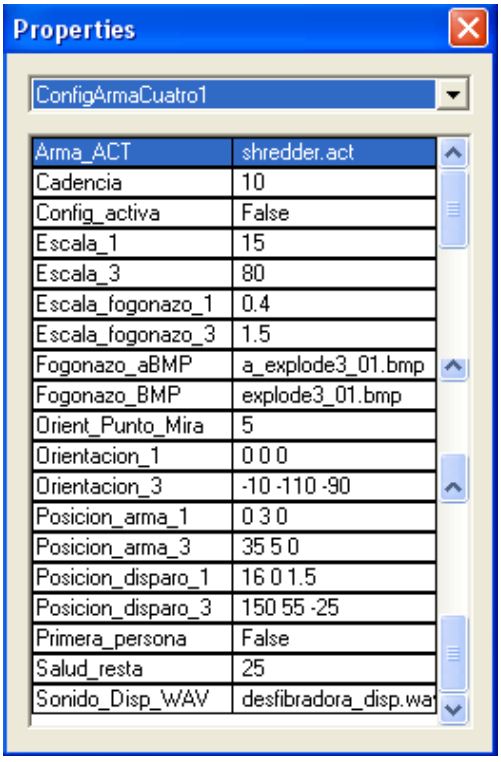
Municion_inicial
Cantidad de munición inicial del arma al recogerla por primera vez.
Se permiten valores de 1 a 100.
Esta cantidad será la que se entregará al jugador al recoger por primera vez esta arma, representada en el escenario con la entidad 'ItemRocket'.

Municion_cargador
Cantidad de munición del cargador para esta arma.
Se permiten valores de 1 a 100.
Esta cantidad será la que se utilizará cuando el jugador recoja el cargador representado en el escenario con la entidad 'ItemRocketAmmo'.

Resta_salud_critica
Cantidad de salud adicional que se restará al enemigo que reciba un impacto directo en su zona crítica, en el caso de que el enemigo tenga definida una zona crítica.
Esta cantidad de salud se sumará a la que ya se haya establecido en el parámetro 'Salud_resta'.
Consulta la entidad BotActorStart para saber como definir zonas críticas en los enemigos.

Comentarios sobre esta entidad:
Consulta la lista de armas disponibles para seleccionar el arma que te interese configurar.
Consulta la lista de proyectiles 3D disponibles para seleccionar el que te interese establecer.
Consulta la lista de partículas disponibles para seleccionar el efecto que te interese establecer.
Consulta la lista de sonidos disponibles para establecer sonidos tanto de disparo como de impacto para el arma.
Puedes utilizar una entidad 'ConfigArmaTres' con valores distintos en cada nivel de tu juego para que el jugador disponga de una arma distinta (como arma número tres) en cada nivel.
Esta entidad la puedes colocar donde quieras mientras se encuentre en el interior del mapa.
Piensa que todas las propiedades que definas para esta arma, serán también asignadas automáticamente para el mismo número de arma de los enemigos.
Las animaciones de las armas sólo funcionan en primera persona y para el jugador, si quieres realizar estos mismos efectos en un jugador en tercera persona o en enemigos, entonces la clave no es animar el arma sino el personaje, diseñando un personaje que incluya una animación 'Shoot' que realice el movimiento de brazos que te interese cuando esté parado y una animación 'Run' con la animación de correr del personaje y al mismo tiempo atacando.
En el caso de un enemigo, exactamente igual y además con el arma ya engancha en su mano derecha.

ConfigArmaCuatro



Permite establecer una serie de parámetros para configurar el arma número cuatro (la que se selecciona pulsando la tecla '4'), permitiendo de esta forma modificar esta arma o crear una completamente nueva.

Parámetros de esta entidad:

Config_activa
Con valor 'True' esta configuración tomará efecto al dar comienzo el nivel.
Con valor 'False' esta configuración no tomará efecto al dar comienzo el nivel.

En un mismo nivel podemos tener más de una entidad 'ConfigArmaCuatro' con parámetros distintos para crear diferentes armas o efectos. Con 'Config_activa' estableceremos cual de esas entidades es la que se tomará como válida al comenzar el nivel, ya que sólo se permite una configuración activa al mismo tiempo, en el resto de entidades 'ConfigArmaCuatro' deberás poner 'False' para este parámetro.

Arma_ACT
Nombre del fichero .ACT que representa el arma que queremos ver.
Este fichero tiene que estar en la carpeta **'armas'** de Entidad 3D.

Sonido_Disp_WAV
Nombre del fichero de sonido .WAV que se reproducirá al disparar el arma.
Este fichero tiene que estar en la carpeta **'wav/armas'** de Entidad 3D.

Fogonazo_BMP
Nombre del fichero BMP que representa el fogonazo que se produce en la boca del cañón al disparar.
Este fichero tiene que estar en la carpeta **'bmp/explode'** de Entidad 3D.

Fogonazo_aBMP
Nombre del fichero BMP (con canal Alpha) que representa el fogonazo.
Este fichero tiene que estar en la carpeta **'bmp/explode'** de Entidad 3D.

Escala_fogonazo_1
Permite establecer el tamaño del fogonazo en primera persona.

Escala_fogonazo_3
Permite establecer el tamaño del fogonazo en tercera persona.

Orientacion_1
Estos tres valores de X Y y Z nos permitirán reorientar el arma (para un juego en primera persona) en caso de que no esté bien orientada. Esto lo podrás observar porque el arma te aparecerá girada apuntando hacia el jugador o hacia los lados, no hacia el frente como debería aparecer.
Estos valores se tomarán en grados, y se permiten desde 1 a 360 tanto negativos como positivos.
En prácticamente el 99% de las veces sólo tendrás que variar el valor Y y en saltos de 90 grados ya que como máximo el arma estará girada de la manera que se ha comentado anteriormente.
Si utilizas un arma que te aparece apuntando hacia arriba, hacia abajo o está ladeada, también tendrás que variar los valores X y Z.
El parámetro 'Orientacion_1' es de los primeros parámetros que debes establecer correctamente para seguir con el resto de parámetros de esta entidad a fin de conseguir colocar adecuadamente el arma.

Orientacion_3
Igual que el parámetro 'Orientacion_1' pero para vistas en tercera persona.
En esta vista en tercera persona intenta que el personaje empuñe el arma con la mano derecha cuando está disparando, y sitúala lo más horizontal posible en esa misma pose (parado y disparando). Del resto de animaciones y poses del personaje no tendrás que preocuparte ya que podrás observar que el personaje siempre mantendrá el arma cogida con la mano derecha de la misma forma que lo hayas establecido en la animación de disparar parado.

Posicion_arma_1
Con estos tres valores de X Y y Z podremos variar la colocación (para un juego en primera persona) del arma frente al jugador.
Variando X la moveremos adelante o atrás dependiendo del signo (+ o -) que utilicemos en el valor.
Variando Y la moveremos arriba o abajo dependiendo del signo (+ o -) que utilicemos en el valor.
Variando Z la moveremos a derecha o izquierda dependiendo del signo (+ o -) que utilicemos en el valor.
Esto es muy útil ya que las armas no son todas iguales ni del mismo tamaño y con la posibilidad de modificar su posición podremos situar el arma en la posición que más nos guste o que nos parezca más realista.
En la mayoría de las ocasiones, sólo bastará modificar el valor X para obtener una posición correcta si has establecido bien el parámetro 'Escala_1', por lo que te aconsejo que antes de ir variando sin control X Y y Z, juegues con los valores X y 'Escala_1' para situar correctamente el arma, en la inmensa mayoría de los casos será suficiente para colocar correctamente el arma frente al jugador.

Posicion_arma_3
Igual que el parámetro 'Posicion_arma_1' pero para vistas en tercera persona.

Orient_Punto_Mira
Orientación hacia el punto de mira en primera persona.
Permite acabar de corregir la orientación del arma para poder girarla hacia el punto de mira.
Este parámetro prácticamente no tendrás que tocarlo nunca si dejas el valor que hay por defecto.

Posicion_disparo_1
Con estos tres valores de X Y y Z podremos variar la posición de donde parte el proyectil disparado en juegos en primera persona.

Normalmente será justo delante del arma, pero puede haber armas en que la salida del proyectil no sea justo al frente.

Posicion_disparo_3
Igual que el parámetro 'Posicion_disparo_1' pero para vistas en tercera persona.

Escala_1
Nos permite establecer el tamaño con el que aparecerá el arma en primera persona. Este valor representa un porcentaje del tamaño original del arma.
Observarás que este valor puede variar bastante de una arma a otra ya que depende de la escala con la que se diseñó y creó el modelo 3D original del arma.

Escala_3
Igual que el parámetro 'Escala_1' pero para vistas en tercera persona.

Cadencia
Disparos por segundo para esta arma. Sólo se admiten valores del 1 al 10. A más valor, más disparos.

Salud_resta
Nivel de salud que se restará al jugador o enemigo que reciba un impacto directo de esta arma.

Primera_persona
Se utiliza para indicar que el arma que se está utilizando es exclusivamente para la vista en primera persona.
Con `True' no se le permitirá al jugador cambiar a otras cámaras que no sea la cámara con vista en primera persona.
Con 'False' se le permitirá al jugador cambiar a las cámara que hayamos permitido mediante la entidad 'Configuracion'.
Poner 'True' es necesario si estamos utilizando un arma en la que se ven las manos o brazos del jugador que sostiene el arma, ya que este modelo 3D no nos serviría para una vista en tercera persona al contener precisamente esas manos o brazos cortados que sólo se ven bien en una vista en primera persona.

AnimDisparo
Nombre de la animación de disparo o ataque en primera persona que se reproducirá cuando el jugador dispare o ataque.
Normalmente esta animación se llama 'Shoot'.
Para que la animación funcione, el fichero .ACT que utilices para esta arma debe contener el nombre de animación que quieres realizar.

AnimDispVel
Establece la velocidad a la que se reproducirá la animación definida en 'AnimDisparo'.
0.1=rápida 0.01=lenta 0.05=media

AnimNDisparo
Nombre de la animación del arma sin disparar o atacar en primera persona que se reproducirá mientras el jugador no dispare o ataque. Normalmente esta animación se llama 'Idle'.
Para que la animación funcione, el fichero .ACT que utilices para esta arma debe contener el nombre de animación que quieres realizar.

AnimNDispVel
Establece la velocidad a la que se reproducirá la animación definida en 'AnimNDisparo'.
0.1=rápida 0.01=lenta 0.05=media

ArmaDeMano
Con `True' simplemente no se descontará munición al disparar, ideal para cuando definas armas cuerpo a cuerpo, pero si consideras que el arma se puede gastar, como un palo, ponlo a 'False'.

Alcance
Alcance del disparo (de 1 a 10000).
Este parámetro te permite definir hasta que punto llega el disparo, de manera que te ofrece la posibilidad de definir un corto alcance (por ejemplo 85) para armas cuerpo a cuerpo como espadas, palos, cuchillos, puños, tuberías, etc..., todo dependerá del diseño 3D que utilices como fichero .ACT
Para que el efecto quede bien no olvides también modificar, sobre todo, estos parámetros:
'Arma_ACT' utiliza un fichero .ACT adecuado a este tipo de arma
'AnimDisparo' con el nombre de la animación de ataque o disparo, normalmente 'Shoot'
'ArmaDeMano' a 'True' para que no descuente munición
'Escala_fogonazo_1' a 0
'Primera_persona' a 'True'

Municion_resta
Cantidad de munición a descontar en cada disparo.
Si pones 0 esta arma tendrá munición infinita.

Municion_inicial

Cantidad de munición inicial del arma al recogerla por primera vez.
Se permiten valores de 1 a 100.
Esta cantidad será la que se entregará al jugador al recoger por primera vez esta arma, representada en el escenario con la entidad 'ItemShredder'.

Municion_cargador
Cantidad de munición del cargador para esta arma.
Se permiten valores de 1 a 100.
Esta cantidad será la que se utilizará cuando el jugador recoja el cargador representado en el escenario con la entidad 'ItemShredderAmmo'.

Resta_salud_critica
Cantidad de salud adicional que se restará al enemigo que reciba un impacto directo en su zona crítica, en el caso de que el enemigo tenga definida una zona crítica.
Esta cantidad de salud se sumará a la que ya se haya establecido en el parámetro 'Salud_resta'.
Consulta la entidad BotActorStart para saber como definir zonas críticas en los enemigos.

Comentarios sobre esta entidad:
Consulta la lista de armas disponibles para seleccionar el arma que te interese configurar.
Consulta la lista de sonidos disponibles para establecer sonidos tanto de disparo como de impacto para el arma.
Puedes utilizar una entidad 'ConfigArmaCuatro' con valores distintos en cada nivel de tu juego para que el jugador disponga de una arma distinta (como arma número cuatro) en cada nivel.
Esta entidad la puedes colocar donde quieras mientras se encuentre en el interior del mapa.
Piensa que todas las propiedades que definas para esta arma, serán también asignadas automáticamente para el mismo número de arma de los enemigos.
Las animaciones de las armas sólo funcionan en primera persona y para el jugador, si quieres realizar estos mismos efectos en un jugador en tercera persona o en enemigos, entonces la clave no es animar el arma sino el personaje, diseñando un personaje que incluya una animación 'Shoot' que realice el movimiento de brazos que te interese cuando esté parado y una animación 'Run' con la animación de correr del personaje y al mismo tiempo atacando.
En el caso de un enemigo, exactamente igual y además con el arma ya engancha en su mano derecha.

ConfigSonidosEne

Config_activa	True
EnemigaDolor_a	daño1a_f.wav
EnemigaDolor_b	daño1b_f.wav
EnemigaDolor_c	daño1c_f.wav
EnemigaEmite_a	emite_ene1a_f.wav
EnemigaEmite_b	emite_ene1b_f.wav
EnemigaMuere_a	muerto1a_f.wav
EnemigaMuere_b	muerto1b_f.wav
EnemigaMuere_c	muerto1c_f.wav
EnemigoDolor_a	daño1a.wav
EnemigoDolor_b	daño1b.wav
EnemigoDolor_c	daño1c.wav
EnemigoEmite_a	emite_ene1a.wav
EnemigoEmite_b	emite_ene1b.wav
EnemigoMuere_a	muerto1a.wav
EnemigoMuere_b	muerto1b.wav
EnemigoMuere_c	muerto1c.wav
Num_enemigo	1

Permite definir todos los sonidos que están asociados a los enemigos.
Los archivos de sonidos deben estar en formato WAV y en la carpeta **'wav'** de Entidad 3D.

Parámetros de esta entidad:
Config_activa
Con valor 'True' esta configuración de sonidos tomará efecto al dar comienzo el nivel.
Con valor 'False' esta configuración de sonidos no tomará efecto al dar comienzo el nivel.
En un mismo nivel podemos tener más de una entidad 'ConfigSonidosEne' con parámetros distintos para crear diferentes grupos de sonidos. Con 'Config_activa' estableceremos cual de esas entidades es la que se tomará como válida al comenzar el nivel, ya que sólo se permite una configuración activa al mismo tiempo para cada enemigo, en el resto de entidades 'ConfigSonidosEne' que hagan referencia a ese enemigo deberás poner 'False' para este parámetro.

Num_enemigo
Número de enemigo al que queremos asignarle todos los sonidos que definiremos en esta entidad 'ConfigSonidosEne'.
Se permiten valores del 1 al 8.

EnemigoDolor_a
Nombre del fichero .WAV del primer sonido de dolor para enemigos.

EnemigoDolor_b
Nombre del fichero .WAV del segundo sonido de dolor para enemigos.

EnemigoDolor_c
Nombre del fichero .WAV del tercer sonido de dolor para enemigos.

EnemigoEmite_a
Nombre del fichero .WAV del primer sonido que emite el enemigo.

EnemigoEmite_b
Nombre del fichero .WAV del segundo sonido que emite el enemigo.

EnemigoMuere_a
Nombre del fichero .WAV del primer sonido de muerte del enemigo.

EnemigoMuere_b
Nombre del fichero .WAV del segundo sonido de muerte del enemigo.

EnemigoMuere_c
Nombre del fichero .WAV del tercer sonido de muerte del enemigo.

EnemigaDolor_a
Nombre del fichero .WAV del primer sonido de dolor para enemigas.

EnemigaDolor_b
Nombre del fichero .WAV del segundo sonido de dolor para enemigas.

EnemigaDolor_c
Nombre del fichero .WAV del tercer sonido de dolor para enemigas.

EnemigaEmite_a
Nombre del fichero .WAV del primer sonido que emite la enemiga.

EnemigaEmite_b
Nombre del fichero .WAV del segundo sonido que emite la enemiga.

EnemigaMuere_a
Nombre del fichero .WAV del primer sonido de muerte de la enemiga.

EnemigaMuere_b
Nombre del fichero .WAV del segundo sonido de muerte de la enemiga.

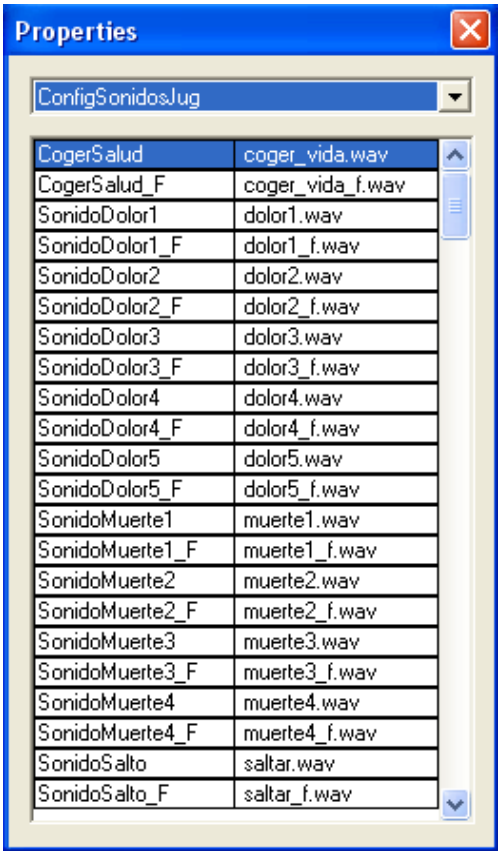
EnemigaMuere_c
Nombre del fichero .WAV del tercer sonido de muerte de la enemiga.

Recuerda que el sexo del enemigo/a lo debes establecer en el parámetro 'Masculino' de la entidad 'BotActorStart' del enemigo correspondiente, y en función de ese parámetro se utilizarán los sonidos masculinos o femeninos que estén definidos en la entidad 'ConfigSonidosEne' de ese/a enemigo/a.

***Nota**: Si quieres que los sonidos que has definido para un enemigo en concreto sean los mismos para todos los niveles del juego, no es necesario tener entidades 'ConfigSonidosEne' para ese enemigo en todos los niveles, con incluir una entidad 'ConfigSonidosEne' para el enemigo en el primer nivel será suficiente.*
Esta entidad la puedes colocar donde quieras mientras se encuentre en el interior del mapa

ConfigSonidosJug

Permite definir todos los sonidos que están asociados al jugador.
Los sonidos deben estar en formato WAV y en la carpeta **'wav'** de Entidad 3D.



Parámetros de esta entidad:

SonidoDolor1
Nombre del fichero .WAV del primer sonido de dolor para el jugador.
SonidoDolor2
Nombre del fichero .WAV del segundo sonido de dolor para el jugador.
SonidoDolor3
Nombre del fichero .WAV del tercer sonido de dolor para el jugador.
SonidoDolor4
Nombre del fichero .WAV del cuarto sonido de dolor para el jugador.
SonidoDolor5
Nombre del fichero .WAV del quinto sonido de dolor para el jugador.
SonidoDolor1_F
Nombre del fichero .WAV del primer sonido de dolor para la jugadora.
SonidoDolor2_F
Nombre del fichero .WAV del segundo sonido de dolor para la jugadora.
SonidoDolor3_F
Nombre del fichero .WAV del tercer sonido de dolor para la jugadora.
SonidoDolor4_F
Nombre del fichero .WAV del cuarto sonido de dolor para la jugadora.
SonidoDolor5_F
Nombre del fichero .WAV del quinto sonido de dolor para la jugadora.
SonidoMuerte1
Nombre del fichero .WAV del primer sonido de muerte para el jugador.
SonidoMuerte2
Nombre del fichero .WAV del segundo sonido de muerte para el jugador.
SonidoMuerte3
Nombre del fichero .WAV del tercer sonido de muerte para el jugador.
SonidoMuerte4
Nombre del fichero .WAV del cuarto sonido de muerte para el jugador.
SonidoMuerte1_F
Nombre del fichero .WAV del primer sonido de muerte para la jugadora.
SonidoMuerte2_F
Nombre del fichero .WAV del segundo sonido de muerte para la jugadora.
SonidoMuerte3_F
Nombre del fichero .WAV del tercer sonido de muerte para la jugadora.
SonidoMuerte4_F
Nombre del fichero .WAV del cuarto sonido de muerte para la jugadora.
CogerSalud
Nombre del fichero .WAV del sonido al recoger salud para el jugador.
CogerSalud_F
Nombre del fichero .WAV del sonido al recoger salud para la jugadora.
SonidoSalto
Nombre del fichero .WAV del sonido al saltar el jugador.
SonidoSalto_F
Nombre del fichero .WAV del sonido al saltar la jugadora.

Recuerda que el sexo del jugador o de la jugadora lo debes establecer en el parámetro 'Jugador_masculino' de la entidad 'DeathMatchStart'.

***Nota**: Sólo es necesaria una entidad '**ConfigSonidosJug**' para todo el juego, con poner esta entidad en el primer nivel será suficiente. Esta entidad la puedes colocar donde quieras mientras se encuentre en el interior del mapa.*

Objetivo

Te permite definir objetivos para tu misión. Objetivos que aparecerán en un panel que el jugador podrá ver al pulsar la tecla 'Borrar hacia atrás':

Parámetros de esta entidad:

Num_objetivo

De 1 a 10.
Número de objetivo.
A medida que vayas añadiendo entidades 'Objetivo' debes asignarles un número diferente.

En_panel

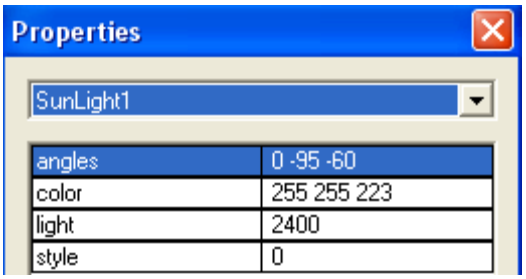
Puede que determinados objetivos no quieras que el jugador los vea nada más comenzar el juego y que posteriormente, durante el transcurso del mismo, los quieras añadir al panel usando el comando 'PAN1'.
Con 'True' en este parámetro el objetivo aparecerá en el panel desde el comienzo del nivel.
Con 'False' en este parámetro, el objetivo estará definido pero no se mostrará en el panel al dar comienzo el nivel.

Descripcion

Texto que aparecerá en el panel para describir este objetivo. Se permite hasta un máximo de 50 caracteres.

Añade tantas entidades 'Objetivo' en tu nivel como objetivos quieras establecer.
Los objetivos no se borran al saltar al siguiente nivel, por lo que de ti depende darlos por finalizados o eliminarlos del panel.
Para añadir objetivos al panel, borrar objetivos del panel o darlos por finalizados, utiliza los comandos 'PAN1', 'PAN0' y 'OBJ1'.
*La imagen que representa el panel es el fichero '**objetivos.bmp**' que se encuentra en la carpeta '**bmp/console/640x480/**'.*
Puedes modificar esta imagen para personalizar tu panel pero no modifiques las dimensiones originales de este .BMP

SunLight



Se utiliza para situar un luz en el escenario que simula la luz del Sol, con rayos siempre paralelos, de manera que nuestro escenario quedará iluminado como si estuviera expuesto a la luz del Sol.

Parámetros de esta entidad:

Style
Se utiliza para producir parpadeos en la luz. Podemos probar con valores '1' '2' o '3'.

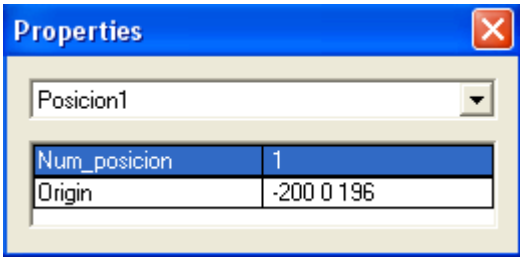
Light
Define el máximo alcance de la luz.

Color
Permite seleccionar el color de la luz.

Angles
Ángulos de inclinación de la luz.
Variando los ángulos X Y y Z nos permitirá orientar la luz hacia cualquier ángulo.

Esta entidad no simula el Sol en sí, sino sólo la luz que desprende, por lo que no busques ver un Sol en el cielo de tu escenario si no es que los has definido tú en forma de ficheros gráficos .BMP en la pestaña 'Sky' de 'World Editor'.
Esta entidad sólo funciona a partir de la habitación en la que está situada la entidad y la luz saldrá de las caras que tengamos definidas como 'Sky'.
ATENCIÓN: Si no asignamos caras 'Sky', la entidad 'SunLight' no emitirá ninguna luz aunque esté bien situada y configurada.

Posicion



Te permite definir posiciones en tu nivel que posteriormente podrás seleccionar desde la entidad 'BotActorStart' para trazar una ruta que seguirán los enemigos o personajes.
Esta entidad también la utiliza el comando 'TELETRANSPORTE'.

Parámetros de esta entidad:

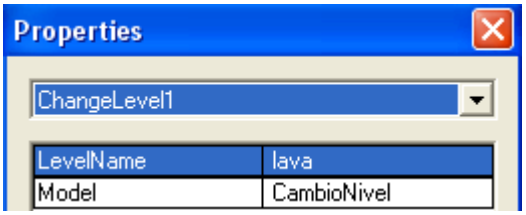
Origin
Son tres cifras (X Y Z) que determinan la posición de esta entidad en el escenario. Estos 3 datos normalmente no se modifican aquí y se deja que se modifiquen según movemos esta entidad (con el ratón) desde 'World Editor'.

Num_posicion
Número de posición (entre 1 y 255).
Son números que deberemos asignar nosotros manualmente.
Este número será el que se utilizará en el comando 'TELETRANSPORTE' para referirse a una posición en concreto para establecer el destino del jugador en un teletransporte.
Estos números de posición siempre han de ser distintos si usamos el comando 'TELETRANSPORTE', aunque si no usamos el comando 'TELETRANSPORTE' en nuestro nivel, puedes ignorar este número de posición y siempre dejar el valor 1 en todas las entidades 'Posicion' que uses.

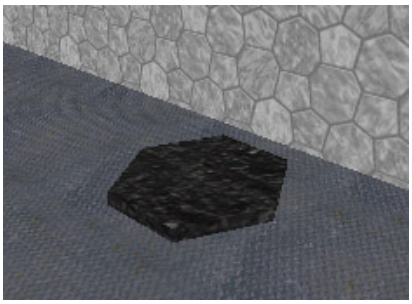
---

*» Añade tantas entidades '**Posicion**' en tu nivel como posiciones quieras establecer para los destinos de tus enemigos o personajes definidos con la entidad '**BotActorStart**' mediante los parámetros 'Destino' de esa entidad, o bien como posiciones para el comando 'TELETRANSPORTE'.*

ChangeLevel



La entidad 'ChangeLevel' sirve para cargar otro nivel cuando el jugador toque un 'model' en concreto.



Parámetros de esta entidad:

LevelName

Nombre del nivel a cargar (por ejemplo **minivel**), sólo el nombre del nivel sin la extensión .BSP. El fichero .BSP de este nivel debe estar en la carpeta 'levels' como el resto de niveles que carga World Editor.

Model

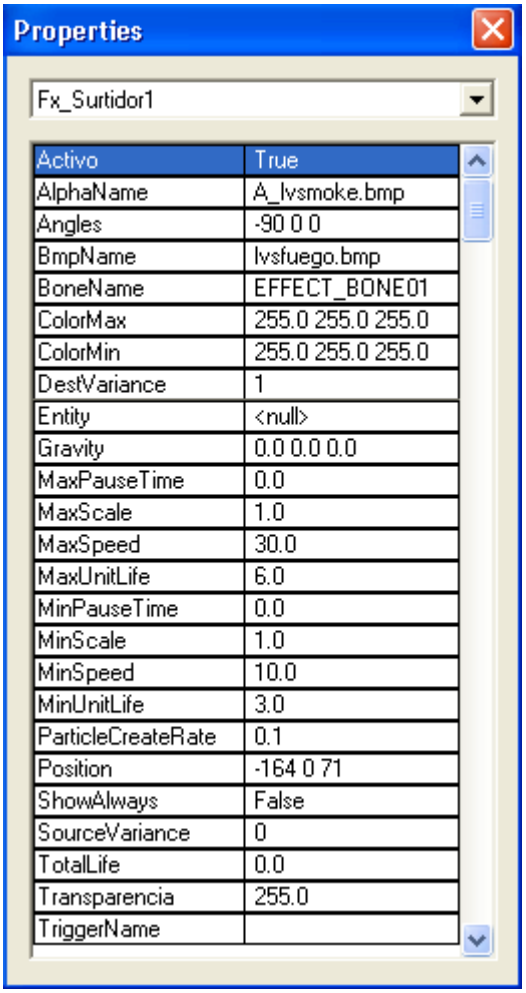
Esta entidad debe estar asociada obligatoriamente a un 'model', y para que funcione correctamente te recomiendo que ese 'model' sea una especie de plataforma, de la altura que quieras, a la que pueda subirse el jugador, al hacerlo pasará al siguiente nivel.

CommandsFile

Nombre de un fichero de comandos .INI que se lanzará justo al iniciar el nuevo nivel.
Al iniciar el nivel, en ocasiones nos puede interesar poner variables con valores concretos, entregar un objeto al jugador, quitarle objetos, darle salud o vidas, presentarle algún mensaje, dar por finalizadas misiones para que no aparezcan en el panel de objetivos, activar temporizadores, etc.., todo esto lo podremos hacer si creamos un fichero .INI en la carpeta '**comandos**' con los comandos necesarios.

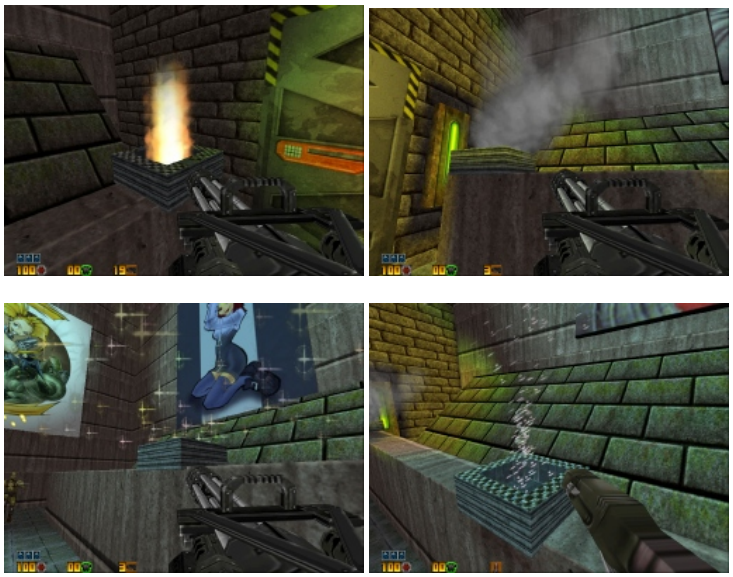
» Ten en cuenta que al saltar a otro nivel, el jugador siempre pierde todos los objetos de su inventario.

Fx_Surtidor



La entidad 'Fx_Surtidor' nos permite definir efectos de partículas como fuego, humo, vapor, chispas, gases, surtidor de agua o cualquier sistema que represente una fuente de partículas.

Estas partículas son en realidad imágenes en formato .BMP con transparencia, que saldrán en varias direcciones, a distintas velocidades y que les podrá afectar la gravedad, permitiéndonos de esta forma definir multitud de efectos realistas, psicodélicos, mágicos o extraterrestres para nuestros juegos.



Parámetros de esta entidad:

Position
Posición del efecto.

Activo
Con 'True' el efecto se pondrá en funcionamiento al dar comienzo el nivel.
Con 'False' el efecto permanecerá inactivo.

BmpName
Nombre del fichero de imagen .BMP que representa la partícula.

AlphaName
Nombre del fichero de imagen .BMP (Alpha) que representa la partícula.

Angles
Variando X Y y Z permite modificar la dirección de emisión de las partículas.

ColorMax
Máximo valor de color para cada partícula. Junto con 'ColorMin' se establece un rango de colores para las partículas, apareciendo de esta forma partículas de diferentes colores aleatoriamente.

ColorMin
Mínimo valor de color para cada partícula. Junto con 'ColorMax' se establece un rango de colores para las partículas, apareciendo de esta forma partículas de diferentes colores aleatoriamente.

DestVariance
Permite que la dirección de emisión de las partículas varíe aleatoriamente. Dejar a 1 si se desea una variación mínima, o elevar ese valor para producir variaciones mayores.

Gravity
Permite modificar el comportamiento de las partículas frente a la gravedad. Valores negativos de Y producen en las partículas una tendencia a caer, mientras que con variaciones en X o Z podremos inclinar las partículas para simular el efecto del viento.

MaxScale
Máximo tamaño de las partículas. Junto con 'MinScale' se establece un rango de tamaños para las partículas, apareciendo de esta forma partículas de diferentes tamaños aleatoriamente.

MinScale
Mínimo tamaño de las partículas. Junto con 'MaxScale' se establece un rango de tamaños para las partículas, apareciendo de esta forma partículas de diferentes tamaños aleatoriamente.

MaxSpeed
Máxima velocidad de las partículas. Junto con 'MinSpeed' se establece un rango de velocidades para las partículas, generando de esta forma partículas con diferentes velocidades aleatoriamente.

MinSpeed
Mínima velocidad de las partículas. Junto con 'MaxSpeed' se establece un rango de velocidades para las partículas, generando de esta forma partículas con diferentes velocidades aleatoriamente.

MaxUnitLife
Máximo tiempo de vida de las partículas. Junto con 'MinUnitLife' se establece un rango de tiempos de vida para las partículas, generando de esta forma partículas con diferentes tiempos de existencia aleatoriamente.

MinUnitLife
Mínimo tiempo de vida de las partículas. Junto con 'MaxUnitLife' se establece un rango de tiempos de vida para las partículas, generando de esta forma partículas con diferentes tiempos de existencia aleatoriamente.

ParticleCreateRate
Establece cada cuantos segundos se genera una nueva partícula. 0.5 es cada medio segundo.

ShowAlways
Con 'False' el efecto sólo se mostrará si la posición de la entidad está a la vista del jugador.
Con 'True' el efecto siempre se mostrará al jugador.

SourceVariance
Permite definir variaciones aleatorias para las posiciones donde se generen las partículas. A mayor valor más dispersas aparecerán las partículas.

TotalLife
Tiempo en segundos que durará el efecto, después del cual el efecto dejará de mostrarse. Poner el valor 0 para que el efecto dure indefinidamente.

Transparencia
Nivel de transparencia del efecto. Entre 0 y 255, siendo más transparente cuanto menor sea el valor.
Normalmente este valor no hay que tocarlo ya que los niveles de transparencia los fija el fichero Alpha que hayamos establecido, pero si aún así queremos conseguir más transparencia sin tocar el fichero Alpha, para por ejemplo crear efectos de vapor, podremos aumentar la transparencia con valores bajos.

BoneName
'Sin función'. Con dejar 'EFFECT_BONE01' es suficiente.

TriggerName
'Sin función'

Entity
'Sin función'

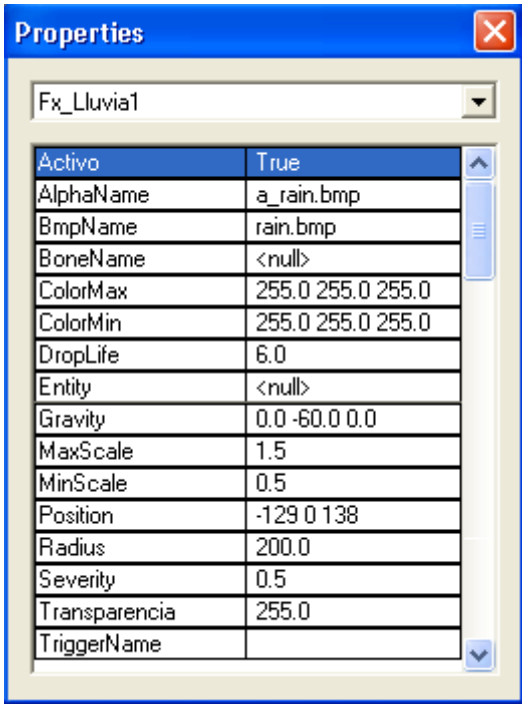
MaxPauseTime
'Sin función'

MinPauseTime
'Sin función'

Model
Permite seleccionar un modelo asociado al efecto.
El efecto quedará enganchado a ese modelo y le seguirá en todos sus movimientos.
Estos modelos son los bloques o conjunto de bloques definidos como 'model' que hayamos establecido en World Editor (pestaña 'models').
Esto nos permite crear multitud de efectos de partículas en movimiento usando la entidad 'Fx_Surtidor' en combinación con 'models' animados. Recuerda que puedes crear 'models' invisibles por lo que al final sólo se vería el efecto de partículas moviéndose por el escenario pero con la ruta que hayas establecido para el 'model' invisible.
¡Atención!, si el efecto supera los límites del escenario dejará de verse y no volverá a reaparecer.

» Variando los diversos parámetros de esta entidad observarás que puedes crear multitud de efectos que se verán con mucha suavidad en tu nivel. Experimenta con esta entidad para conocer todas sus posibilidades.

Fx_Lluvia



Las entidades 'Fx_Lluvia' nos permiten crear lluvia en las zonas de nuestro escenario que las situemos.
De hecho también podemos hacer que caiga nieve o lo que queramos con sólo cambiar la textura BMP de las gotas.

Parámetros de esta entidad:

Position
Fija la posición de salida de las gotas.

Activo
Con 'True' la lluvia se pondrá en funcionamiento al dar comienzo el nivel.
Con 'False' la lluvia permanecerá inactiva.

BmpName
Nombre del fichero de imagen .BMP que representa una gota.

AlphaName
Nombre del fichero de imagen .BMP (Alpha) que representa una gota.

Gravity
Permite modificar el comportamiento de la lluvia frente a la gravedad. Los valores negativos de Y hacen que las gotas caigan, mientras que con variaciones en X y Z podremos inclinar la lluvia para simular el efecto del viento.

Radius
Radio de acción de la lluvia.

Severity
Intensidad de la lluvia. 0.0 poca, 1.0 mucha

DropLife
Vida de cada gota (en segundos).
Permite establecer cuanto tarda en desaparecer la gota una vez ha salido de su origen. Lo ideal es que desaparezca cuando ha llegue al suelo, por lo que este valor dependerá de la altura a la que sitúes la entidad 'Fx_Lluvia'.

ColorMax
Máximo valor de color para cada gota. Junto con 'ColorMin' se establece un rango de colores para las gotas, apareciendo de esta forma gotas de diferentes colores aleatoriamente.

ColorMin
Mínimo valor de color para cada gota. Junto con 'ColorMax' se establece un rango de colores para las gotas, apareciendo de esta forma gotas de diferentes colores aleatoriamente.

Transparencia
Nivel de transparencia de las gotas. Entre 0 y 255, siendo más transparente cuanto menor sea el valor.

MaxScale
Máximo tamaño de las gotas. Junto con 'MinScale' se establece un rango de tamaños para las gotas, apareciendo de esta forma gotas de diferentes tamaños aleatoriamente.

MinScale
Mínimo tamaño de las gotas. Junto con 'MaxScale' se establece un rango de tamaños para las gotas, apareciendo de esta forma gotas de diferentes tamaños aleatoriamente.

BoneName
'Sin función'. Con dejarlo como está es suficiente.

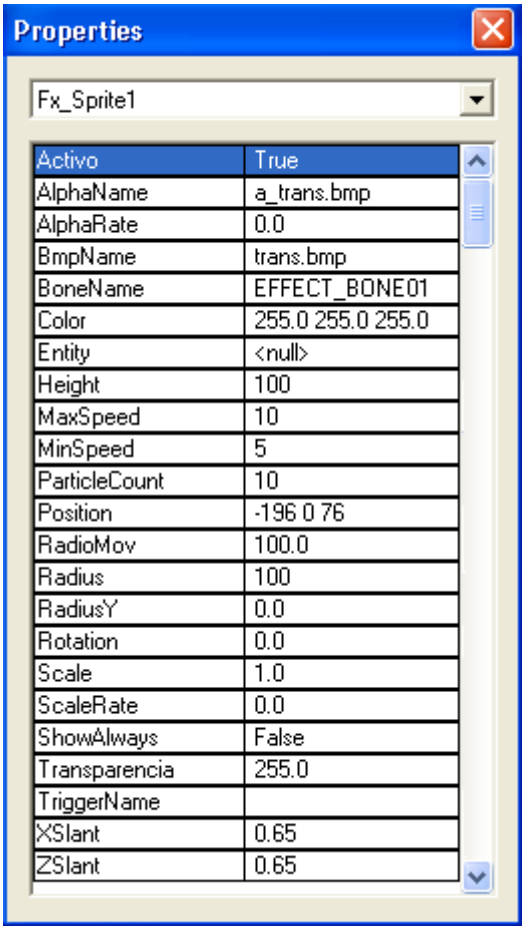
TriggerName
'Sin función'

Para crear un efecto de lluvia en una zona extensa es más recomendable, para mejorar el rendimiento del juego, situar varias entidades 'Fx_Lluvia' de un radio medio que no una sola entidad con un radio muy grande.
Cambiando el nombre del fichero 'BmpName' por el de '**nieve.bmp**' y en 'AlphaName' también por el de '**nieve.bmp**' obtendremos nieve en lugar de agua. En realidad puedes cambiar los ficheros .BMP para hacer que llueva lo que quieras (ranas, estrellas, bolas, flores, etc..) todo dependerá del diseño de ficheros BMP que utilices. En principio sólo existen los dos diseños que se han comentado (agua y nieve) pero si creas alguno nuevo sólo deberás copiar los ficheros .BMP que crees en la carpeta 'BMP/FX' y asignar esos nombres de ficheros .BMP en los parámetros 'BmpName' y 'AlphaName'.

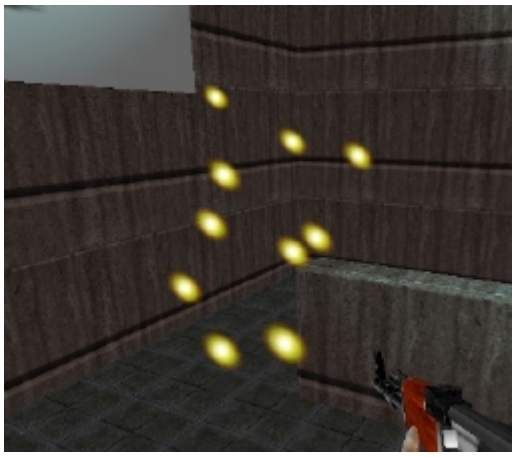
---

*» Los ficheros .BMP que utilices para esta entidad siempre han de estar en la carpeta '**BMP/FX**'.*
» Para crear el efecto de sonido de la lluvia al caer, tendrás que utilizar el comando 'SODF'.
» Si utilizas 'Fx_Lluvia' para crear filtraciones de agua en el interior de un edificio, es más conveniente que uses la entidad 'ElectricBolt' en lugar del comando 'SODF' para crear el efecto de sonido del agua al caer, ya que de esta forma el sonido se oirá sólo en la zona que esté cayendo el agua.

Fx_Sprite



Las entidades 'Fx_Sprite' nos permiten crear grupos de partículas suspendidas en el aire o en el suelo que tendrán un movimiento continuo.



Parámetros de esta entidad:

Position
Posición del efecto.

Activo
Con 'True' el efecto se pondrá en funcionamiento al dar comienzo el nivel.
Con 'False' el efecto permanecerá inactivo.

Color
Color de las partículas.

BmpName
Nombre del fichero de imagen .BMP que representa la partícula.

AlphaName
Nombre del fichero de imagen .BMP (Alpha) que representa la partícula.

Scale
Tamaño de las partículas.

ParticleCount
Cantidad de partículas que intervendrán en el efecto.

Radius
Radio horizontal de colocación de las partículas.

RadiusY
Radio vertical de colocación de las partículas.

MinSpeed
Mínima velocidad de las partículas. Junto con 'MaxSpeed' se establece un rango de velocidades para las partículas, generando de esta forma partículas con diferentes velocidades aleatoriamente.

MaxSpeed
Máxima velocidad de las partículas. Junto con 'MinSpeed' se establece un rango de velocidades para las partículas, generando de esta forma partículas con diferentes velocidades aleatoriamente.

Rotation
Velocidad de rotación de las partículas (entre -3.14 y 3.14). Dejar a 0 si no se desea rotación.

ScaleRate
Velocidad de reducción del tamaño de las partículas. El valor ha de ser entre 0 y lo que hayamos puesto en 'Scale'. Permite que el tamaño de las partículas vaya reduciéndose a medida que el efecto se muestra. Dejar a 0 si no se desea que se reduzca.

Transparencia
Nivel de transparencia de las partículas. Entre 0 y 255, siendo más transparente cuanto menor sea el valor.

AlphaRate
Velocidad de reducción de la transparencia de las partículas. El valor ha de ser entre 0 y lo que hayamos puesto en 'Transparencia'.
Permite que el nivel de transparencia de las partículas vaya reduciéndose a medida que el efecto se muestra. Dejar a 0 si no de desea que se reduzca el nivel de transparencia.

ShowAlways
Con 'False' el efecto sólo se mostrará si la posición de la entidad está a la vista del jugador.
Con 'True' el efecto siempre se mostrará al jugador.

RadioMov
Radio de acción de las partículas.
Permite establecer el alcance máximo de las partículas en su movimiento.

BoneName
'Sin función'. Con dejar 'EFFECT_BONE01' es suficiente.

Height
'Sin función'

XSlant
'Sin función'

ZSlant
'Sin función'

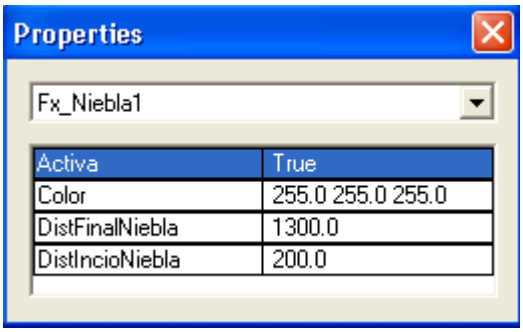
Entity
'Sin función'

TriggerName
'Sin función'

---

*» Los ficheros .BMP que utilices para esta entidad siempre tienen que estar en la carpeta **'BMP/FX'**.*

Fx_Niebla



La entidad 'Fx_Niebla' nos permite crear en nuestro nivel un efecto de niebla en la distancia.

Parámetros de esta entidad:

Activa
Con 'True' la niebla se podrá ver desde el momento que dé comienzo el nivel.
Con 'False' la niebla permanecerá inactiva y no se verá al dar comienzo el nivel.

Color

Permite seleccionar el color de la niebla.

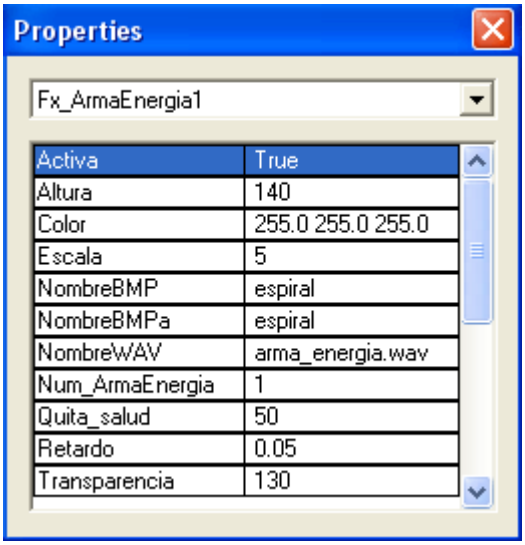
DistIncioNiebla
Distancia del comienzo de la niebla. Entre 1 y 5000.

DistFinalNiebla
Distancia del final de la niebla. Entre 1 y 5000.

RenderTrasNiebla
Con 'False', el motor 3D no calculará nada de lo que haya detrás de la niebla, aunque sólo a nivel de escenario, el resto de elementos como personajes y enemigos se seguirá procesando de forma normal.
Con 'True', el motor 3D seguirá calculando, aunque no se vea, todo lo que hay detrás de la niebla.
Con el uso de 'False' en este parámetro se pueden conseguir rendimientos de velocidad muy altos y con un aspecto visual aceptable si lo combinamos con distancias razonables para la niebla, aunque si usamos este efecto en exteriores, en las zonas marcadas como 'Sky' no tendrá efecto y el cielo se seguirá calculando y visualizando, por lo que si no te interesa que se vea esta combinación de niebla en la distancia pero con un cielo que se ve perfectamente, tendrás que poner 'True' en este parámetro.

» Sólo es necesaria una entidad '**Fx_Niebla**' en el nivel para obtener el efecto de niebla de fondo en todo ese nivel. La entidad puede estar situada en cualquier parte del nivel mientras se encuentre en el interior del mapa.
» No pienses en esta niebla sólo como un efecto atmosférico para exteriores, sino como un recurso que incluso puede dar muy buenos resultados visuales en interiores si usamos las distancias adecuadas y los colores de niebla adecuados.
» Consulta la sección de los comandos 'NIE1' y 'NIE0' para saber como activar y desactivar este efecto de niebla durante el transcurso del juego.

Fx_ArmaEnergia



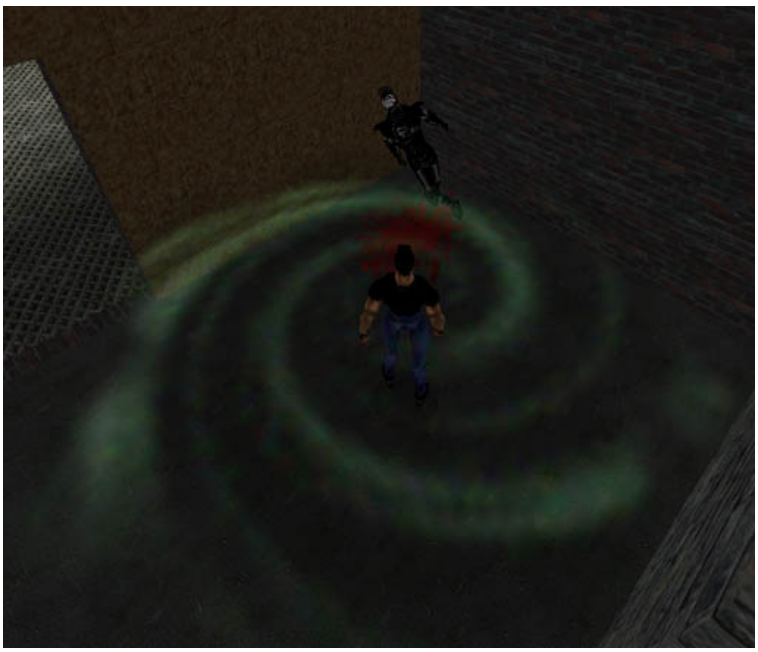
Permite definir las armas de energía que podrá usar el jugador.

Parámetros de esta entidad:

Activa
Con 'True', esta entidad se tendrá en cuenta en la compilación del nivel con 'World Editor' y definirá la arma correspondiente.
Con 'False', esta entidad no se tendrá en cuenta en la compilación.

Num_ArmaEnergia
Número de arma de energía (de 1 a 20).
Son números que deberemos asignar nosotros manualmente.
Este número será el que se utilizará en el comando 'ARMAENERGIA' para referirse a una arma en concreto para poder activarla.

NombreBMP
Nombre de la animación del efecto de la arma.
Con este nombre, Entidad 3D buscará en la carpeta **'bmp\fx'** una secuencia de imágenes .BMP que utilizará para presentar una animación del efecto, hasta un máximo de 32 imágenes.



Por ejemplo, si ponemos el nombre **onda** en este parámetro, Entidad 3D buscará los ficheros:
bmp\fx\onda1.bmp
bmp\fx\onda2.bmp
bmp\fx\onda3.bmp
bmp\fx\onda4.bmp
bmp\fx\onda5.bmp
.
.
.
bmp\fx\onda32.bmp
No tienen porque existir los 32 ficheros, sólo debe haber los necesarios para que funcione el efecto.
Las imágenes deben ser todas del mismo tamaño, y de alguno de los siguientes tamaños: 16x16, 32x32, 64x64, 128x128 o 256x256.
Las imágenes pueden ser RGB o de 256 colores.

NombreBMPa
Nombre del fichero .BMP (con canal Alpha) que representa la secuencia del efecto del arma, pero con canales Alpha de transparencia para crear el efecto llamado Alpha Blending.
Estas imágenes deben tener las mismas

características que el parámetro 'NombreBMP' pero con blancos, negros y grises para establecer los niveles de transparencia de cada una de las imágenes de la secuencia.
Por ejemplo, si ponemos el nombre **onda_a** en este parámetro, Entidad 3D buscará los ficheros:
bmp\fx\onda_a1.bmp
bmp\fx\onda_a2.bmp
bmp\fx\onda_a3.bmp
bmp\fx\onda_a4.bmp
bmp\fx\onda_a5.bmp
.
bmp\fx\onda_a32.bmp
Si no tienes o no quieres crear ficheros Alpha para un efecto en concreto, puedes indicar en 'NombreBMPa' el mismo nombre que has indicado en 'NombreBMP', para que el programa utilice las mismas imágenes del efecto para crear transparencias.
En muchas ocasiones este método de transparencias suele dar buenos resultados.

NombreWAV
Nombre del fichero de sonido .WAV que se reproducirá al usar esta arma.
Este fichero tiene que estar en la carpeta **'wav/armas'** de Entidad 3D.

Retardo
Tiempo (en segundos) de intervalo entre cada imagen BMP de la secuencia del efecto.
Permite controlar la velocidad del efecto.

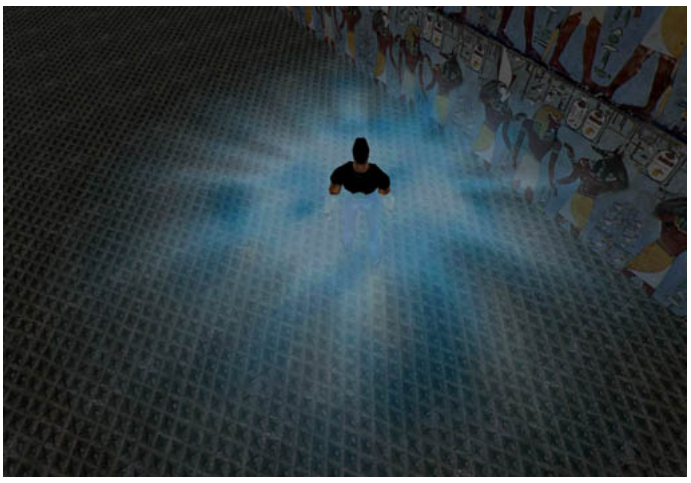
Escala
Tamaño del efecto.
Permite definir el tamaño del efecto y por lo tanto el alcance que tendrá la arma.

Quita_salud
Cantidad de salud a restar a los enemigos (entre 1 y 100).
Establece la cantidad de salud que se resta a cada enemigo que es alcanzado por la arma.

Altura
Altura del efecto.
Permite definir a que altura, medida desde el suelo, se deproducirá el efecto de la arma alrededor del jugador, siendo aproximadamente 140 la altura de la cintura del jugador.

Transparencia
Nivel de transparencia del efecto. Entre 0 y 255, siendo más transparente cuanto menor sea el valor.

Color
Color del efecto
Te permite seleccionar un color que se mezclará con la animación para dar tonalidades adicionales a toda la secuencia del efecto.

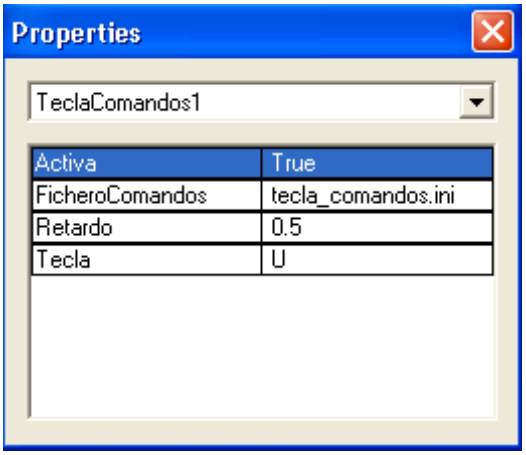


¿ Qué son las armas de energía ?:

Son armas que no se usan de la misma manera que se usan las cuatro armas normales que se seleccionan con las teclas del 1 al 4 y se disparan con la tecla o botón de disparo. Como has visto en esta sección, las armas de energía se definen con entidades '**Fx_ArmaEnergia**' y se disparan con el comando 'ARMAENERGIA', por lo que nosotros, como creadores del juego, debemos incluir esos comandos en ficheros de comandos .INI para producir el efecto de que el jugador dispara esas armas de energía.
El efecto es una secuencia animada que se produce alrededor del jugador y en círculo, provocando daño a todos los enemigos que alcanza.
Al estar definido el efecto por múltiples imágenes BMP, nos permite crear nuestros propios efectos, como por ejemplo: rayos, espirales, gases, fuego, partículas, ondas, etc...
Estas armas están pensadas para ser usadas asociadas a teclas usando entidades 'TeclaComandos' o bien usando acciones del menú de acciones del jugador, pero en realidad se pueden activar usando cualquier mecanismo que se te ocurra combinando comandos y condiciones.

» Una arma de energía definida en un nivel sólo sirve para ese nivel.
» Consulta el comando 'ARMAENERGIA' para ver como se disparan este tipo de armas.
» Consulta la entidad BotActorStart para saber como poder inmunizar a los enemigos frente a estas armas.

TeclaComandos



La entidad 'TeclaComandos' sirve para asignar a ciertas teclas el lanzamiento de ficheros de comandos .INI, por lo que nos permitirá ofrecer al jugador la posibilidad de tener diversas funciones según la tecla que pulse.

Parámetros de esta entidad:

FicheroComandos
Nombre de un fichero de comandos .INI que se lanzará cuando el jugador pulse la tecla que hemos establecido en el parámetro 'Tecla'. Recuerda que el fichero de comandos .INI debe estar en la carpeta '**comandos**' de Entidad 3D.

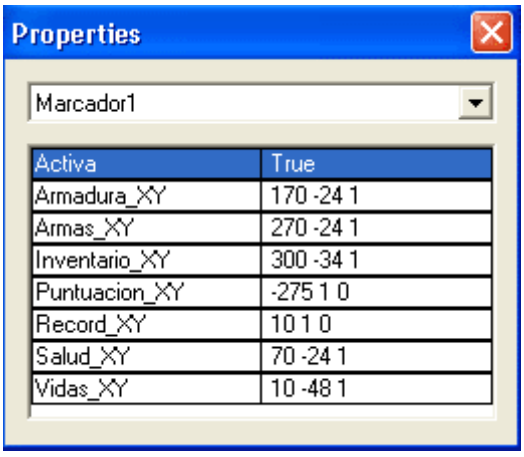
Tecla
Tecla que deberá pulsar el jugador para lanzar el fichero de comandos que hayamos establecido en 'FicheroComandos'.
Debe ser un solo carácter exceptuando algunas teclas especiales que deberemos indicar usando alguna de las siguientes palabras clave:
F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, PAUSE, SCROLL, SNAPSHOT, INSERT, DELETE, PRIOR, NEXT, END, HOME, CAPITAL, CONTROL, SHIFT, ALT, TAB, BACK, MBUTTON, IZQUIERDA, DERECHA, ARRIBA, ABAJO, NUMPAD0, NUMPAD1, NUMPAD2, NUMPAD3, NUMPAD4, NUMPAD5, NUMPAD6, NUMPAD7, NUMPAD8, NUMPAD9, NUMPAD*, NUMPAD/, NUMPAD+, NUMPAD-, NUMPAD., IZQSHIFT, DERSHIFT, IZQCONTROL, DERCONTROL

Retardo
Tiempo, en segundos, en permitir volver a usar la tecla.

Activa
Con valor 'True' la configuración de esta tecla tomará efecto al dar comienzo el nivel.
Con valor 'False' esta configuración no tomará efecto al dar comienzo el nivel.

» Añade tantas entidades 'TeclaComandos' en el primer nivel como teclas quieras configurar hasta un máximo de 16. Estas configuraciones se seguirán manteniendo en todos los niveles del juego.
» Esta entidad puede estar situada en cualquier parte del nivel mientras se encuentre en el interior del mapa.

Marcador



Esta entidad permite situar cualquier elemento del marcador de Entidad 3D (vidas, salud, armadura, armas, inventario, puntuación y récord) en cualquier posición de la pantalla. También se pueden ocultar los elementos que se seleccionen.

Parámetros de esta entidad:

Activa
Con valor 'True' esta configuración del marcador tomará efecto al dar comienzo el nivel.
Con valor 'False' esta configuración no tomará efecto al dar comienzo el nivel.

Vidas_XY
'X' 'Y': Coordenadas **x y** que establecen la posición en pantalla del indicador de vidas del jugador. Muestra un pequeño icono por cada vida que tenga el jugador.
'Z': Con valor 1 el indicador de vidas se presentará en pantalla, mientras que con valor 0 permanecerá oculto.

VidasNum_XY
'X' 'Y': Coordenadas **x y** que establecen la posición en pantalla del contador de vidas del jugador.
'Z': Con valor 1 el contador de vidas se presentará en pantalla, mientras que con valor 0 permanecerá oculto.

Salud_XY
'X' 'Y': Coordenadas **x y** que establecen la posición en pantalla del indicador de salud.
'Z': Con valor 1 el indicador de salud se presentará en pantalla, mientras que con valor 0 permanecerá oculto.

Armadura_XY
'X' 'Y': Coordenadas **x y** que establecen la posición en pantalla del indicador de armadura.
'Z': Con valor 1 el indicador de armadura se presentará en pantalla, mientras que con valor 0 permanecerá oculto.

Armas_XY
'X' 'Y': Coordenadas **x y** que establecen la posición en pantalla del indicador de armas y munición.
'Z': Con valor 1 el indicador de armas y munición se presentará en pantalla, mientras que con valor 0 permanecerá oculto.

Inventario_XY
'X' 'Y': Coordenadas **x y** que establecen la posición en pantalla del inventario de objetos.
'Z': Con valor 1 el inventario de objetos se presentará en pantalla cuando el jugador pulse la tecla de inventario, mientras que con valor 0 permanecerá oculto todo el tiempo.
Con 'Z' a 1 el jugador podrá hacer aparecer y desaparecer el panel usando la tecla '**i**'.

Puntuacion_XY
'X' 'Y': Coordenadas **x y** que establecen la posición en pantalla del indicador de puntuación.
'Z': Con valor 1 el indicador de puntuación se presentará en pantalla, mientras que con valor 0 permanecerá oculto.

Record_XY
'X' 'Y': Coordenadas **x y** que establecen la posición en pantalla del indicador de récord.
'Z': Con valor 1 el indicador de récord se presentará en pantalla, mientras que con valor 0 permanecerá oculto.

PanelObjeto_XY
'X' 'Y': Coordenadas **x y** que establecen la posición en pantalla del panel de objeto.
Valores **0** para las coordenadas **'Y'** y '**X**', centrará el panel de objeto en medio de la pantalla.
'Z': Con valor 1 el panel de objeto se presentará en pantalla, mientras que con valor 0 permanecerá oculto.

Acciones_XY
'X' 'Y': Coordenadas **x y** que establecen la posición en pantalla del menú de acciones.
Valores **0** para las coordenadas **'Y'** y '**X**', centrará el menú de acciones en medio de la pantalla.
'Z': Con valor 1 el menú de acciones se presentará en pantalla, mientras que con valor 0 permanecerá oculto.
Con 'Z' a 1 el jugador podrá hacer aparecer y desaparecer el menú usando la tecla '**i**'.

Dinero_XY
'X' 'Y': Coordenadas **x y** que establecen la posición en pantalla del indicador de dinero del jugador.
'Z': Con valor 1 el indicador de dinero se presentará en pantalla, mientras que con valor 0 permanecerá oculto.

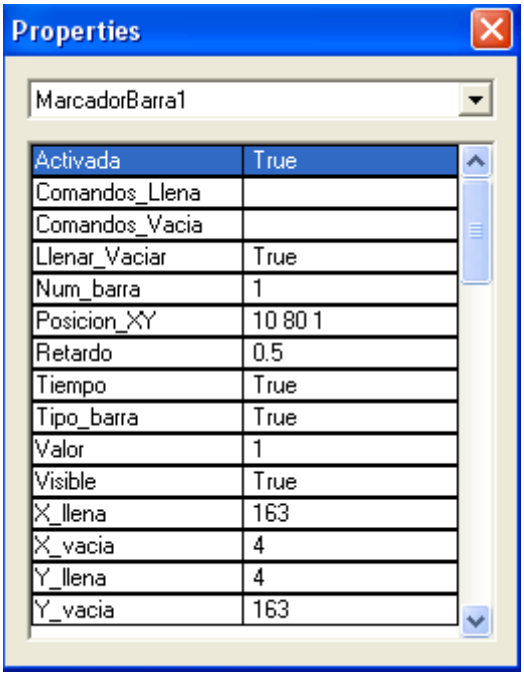
Un valor **'X' positivo** en cualquier coordenada indicará que la coordenada de **x** se contará desde el margen izquierdo de la pantalla.
Un valor **'X' negativo** en cualquier coordenada indicará que la coordenada de **x** se contará desde el margen derecho de la pantalla.

Un valor **'Y' positivo** en cualquier coordenada indicará que la coordenada de **y** se contará desde la parte superior de la pantalla.
Un valor **'Y' negativo** en cualquier coordenada indicará que la coordenada de **y** se contará desde la parte inferior de la pantalla.

---

*» Consulta el apartado titulado '**Coordenadas x y relativas a los bordes de pantalla**' de la sección del comando 'MSJ1' para saber como funciona el sistema de posicionamiento de elementos en pantalla usando coordenadas **x y** positivas y/o negativas.*
*» Sólo es necesaria una entidad '**Marcador**' en el primer nivel para modificar la posición de los elementos del marcador en todos los niveles de tu juego. La entidad puede estar situada en cualquier parte del nivel mientras se encuentre en el interior del mapa.*

MarcadorBarra



Esta entidad permite definir barras de estado como parte del sistema de marcadores que tengamos en nuestro juego.
Permite usar ficheros gráficos BMP para crear nuestras propias barras de estado usando ficheros BMP que hayamos podido diseñar.

Parámetros de esta entidad:

Activada

Con valor 'True' esta barra estará activada al dar comienzo el nivel. Esto significa que si está a 'True' el parámetro 'Tiempo', esta barra comenzará a llenarse/vaciarse desde que comience el nivel.
Con valor 'False' esta barra no estará activada al dar comienzo el nivel, aunque durante el transcurso del juego la podremos activar con el comando 'BARRAACTIVAR'.
El comando 'BARRADESACTIVAR' nos permitirá desactivar una barra durante el transcurso del juego.

Num_barra

Número de barra (entre 1 y 10).
Son números que deberemos asignar nosotros manualmente.
Este número será el que se utilizará en los comandos 'BARRAACTIVAR', 'BARRADESACTIVAR', 'BARRAVISIBLE', 'BARRAINVISIBLE' y 'BARRAVALOR', para referirse a una barra en concreto.

Visible

Permite indicar si la barra será visible desde el mismo instante que de comienzo el nivel o no.
Con valor 'True' será visible desde el comienzo del nivel, aunque el comando 'BARRAINVISIBLE' te permitirá hacer invisible la barra durante el transcurso del juego.
Con valor 'False' será invisible desde el comienzo del nivel, aunque el comando 'BARRAVISIBLE' te permitirá hacer visible la barra durante el transcurso del juego.

Posicion_XY

Permite indicar la posición donde aparecerá la barra en pantalla.
Aunque en este parámetro aparecen tres cajas para rellenar (X Y y Z), sólo debes usar X Y.
Un valor **'X' positivo** indicará que la coordenada de **X** se contará desde el margen izquierdo de la pantalla.
Un valor **'X' negativo** indicará que la coordenada de **X** se contará desde el margen derecho de la pantalla.
Un valor **'Y' positivo** indicará que la coordenada de **Y** se contará desde la parte superior de la pantalla.
Un valor **'Y' negativo** indicará que la coordenada de **Y** se contará desde la parte inferior de la pantalla.

Tipo_barra

Aquí debemos indicar que tipo de orientación tiene la barra que queremos usar, horizontal o vertical.
Con valor 'True', horizontal.
Con valor 'False', vertical.

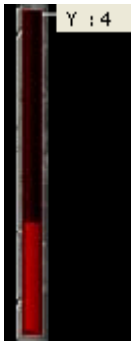
X_llena

Sólo usar para barras horizontales.
Es un valor que debemos indicar dependiendo del diseño de barra que utilicemos. Debe indicar que coordenada X de la barra, es la que se considerará como barra completamente llena cuando el llenado de la barra alcance esa posición.
Si utilizamos la barra número 1 (parámetro 'Num_barra' con valor 1), que es la barra por defecto, este valor es siempre **163**, tal y como podemos ver a continuación si editamos el fichero BMP de esta barra ('barra1a.bmp') con un programa de edición 2D, considerando 0 la coordenada más a la izquierda:

X_vacia

Sólo usar para barras horizontales.
Lo mismo que el parámetro anterior pero para indicar la X de la barra vacía:

Y_llena
Sólo usar para barras verticales.
Lo mismo que el parámetro 'X_llena' pero para indicar la Y de la barra llena:



Y_vacia
Sólo usar para barras verticales.
Lo mismo que el parámetro 'X_vacia' pero para indicar la Y de la barra vacía:

Valor
Permite indicar un valor que representará el nivel de llenado inicial de la barra. De esta forma una barra puede comenzar completamente vacía, llena o con el nivel que queramos.
Como mínimo puede valer **1** y como máximo depende del tamaño de la barra.
Este valor podremos cambiarlo durante el transcurso del juego usando el comando 'BARRAVALOR'.

Llenar_Vaciar
Sirve para indicar si la barra se llenará o se vaciará con el paso del tiempo.
Con valor 'True' se llenará.
Con valor 'False' se vaciará.
Sólo funcionará si la barra está activada y el parámetro 'Tiempo' está a 'True'.

Tiempo
Con valor 'True', la barra se llenará/vaciará (depende del parámetro 'Llenar_Vaciar') con el paso del tiempo.
Con valor 'False', la barra no se llenará/vaciará con el paso del tiempo.

Retardo
Permite controlar la velocidad de llenado/vaciado de la barra cuando está activa y el parámetro 'Tiempo' está a 'True'.
El valor de este parámetro ha de ser un tiempo expresado en segundos.

Comandos_Llena
Nombre del fichero de comandos .INI que se procesará cuando la barra se llene por completo.
Deja en blanco este parámetro si quieres que no se procese ningún fichero de comandos.

Comandos_Vacia
Nombre del fichero de comandos .INI que se procesará cuando la barra se vacíe por completo.
Deja en blanco este parámetro si quieres que no se procese ningún fichero de comandos.

Ficheros BMP que definen las barras:
Para definir cada barra puedes usar imágenes RGB o de 256 colores. Se necesitan dos imágenes para cada barra.
Si usas imágenes RGB la calidad gráfica de la barra siempre será mejor que con 256 colores, pero tendrás que crear los dos ficheros Alpha adicionales para las transparencias de cada imagen:
En caso de usar imágenes BMP de 256 colores los dos ficheros necesarios serían:
Imagen tipo '**a**': Un fichero BMP con la imagen de la barra completa (con su marco, figuras, etc...), pero con la barra de llenado completamente vacía. Las zonas transparentes deberán estar pintadas como siempre con el color 255 (el último color de la paleta de colores).

Imagen tipo '**b**': Un fichero BMP del mismo tamaño que el anterior pero sólo mostrando la barra de nivel completamente llena y colocada en su sitio correcto. Todo lo que no sea la barra llena, píntalo de color transparente.
Ejemplo de una barra de 256 colores:
Fichero BMP de imagen tipo '**a**':

Fichero BMP de imagen tipo '**b**':

Los ficheros BMP deben estar en la carpeta **'bmp\barras'** y deben tener un nombre en concreto en función de: el nombre del primer nivel del juego donde aparecerá la barra, el número de barra que queremos definir (que hemos indicado en el parámetro '**Num_barra**') y de si son imágenes de tipo **'a'** o **'b'**.
En el caso del ejemplo anterior, si la quisiéramos definir como la barra número 2, para un juego cuyo primer nivel se llama 'desierto', los nombres de los ficheros tendrían que ser exactamente estos:
desierto_barra2a.bmp (como imagen '**a**').
desierto_barra2b.bmp (como imagen '**b**').
... si fueran para definir la barra número 5 pues serían así:
desierto_barra5a.bmp (como imagen '**a**').
desierto_barra5b.bmp (como imagen '**b**').

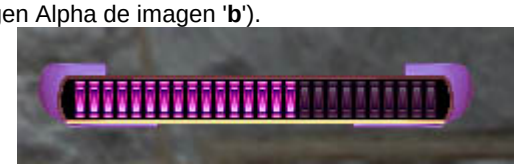
Ejemplo de una barra RGB (24bits de color):
Fichero BMP de imagen tipo '**a**':

Fichero BMP de imagen tipo '**b**':

Fichero BMP Alpha para imagen '**a**':

Fichero BMP Alpha para imagen '**b**':

... y sus nombres como barra número 3 en un juego cuyo primer nivel se llamase 'desierto' tendrían que ser:
desierto_barra3a.bmp (como imagen '**a**').
desierto_barra3b.bmp (como imagen '**b**').
desierto_a_barra3a.bmp (como imagen Alpha de imagen '**a**').
desierto_a_barra3b.bmp (como imagen Alpha de imagen '**b**').



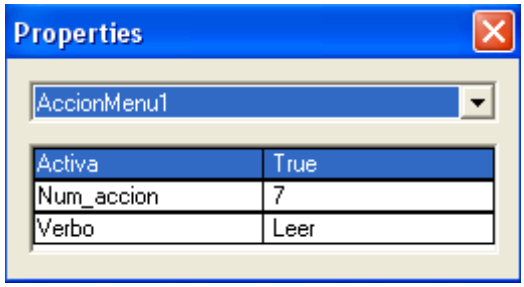
Si de entrada no quieres complicarte la vida con nombres de ficheros .BMP para definir nuevos diseños de barras para tus juegos, puedes usar la barra número 1 que existe por defecto en Entidad 3D, es de tipo horizontal y sirve para cualquier nombre de nivel. Simplemente añade una entidad '**MarcadorBarra**' a tu nivel.
Si quieres usar esa misma barra por defecto, pero en su diseño vertical, usa la barra número 2. En este caso recuerda poner a 'False' (para barra vertical) el parámetro 'Tipo_barra'.

¿ Para qué pueden servir estas barras ?:
Controlar el oxígeno que le queda al jugador al entrar en una cámara donde debe completar diversas tareas.
Controlar el tiempo que falta para que explote algo.
Indicador de dañó que sufre un reactor por los disparos del jugador.
Indicadores de niveles de energía de un motor, el jugador debe conseguir llenarlos por completo para usarlo.
Indicador de 'stress' del jugador.
Controlar el tiempo que una puerta permanece abierta para que el jugador escape por ella.
Controlar el tiempo de que dispone el jugador para liberar a los rehenes.
Presentar el nivel de daño que vaya sufriendo un enemigo final mientras es atacado por el jugador.
Controlar el tiempo de que dispone el jugador para conseguir un antídoto que le salve.

---

*» Las barras definidas con entidades '**MarcadorBarra**' en el primer nivel de tu juego, las podrás usar sin problemas en los siguientes niveles sin tener que volver a situar nuevas entidades '**MarcadorBarra**' en esos niveles.*
» De ti depende lo que hay que hacer cuando las barras se llenen o se vacíen por completo usando ficheros de comandos. Ten un cuenta que las barras no se vuelven a llenar ni a vaciar automáticamente cuando alcanzan su final.
*» Las entidades '**MarcadorBarra**' puedes situarlas en cualquier parte del nivel mientras se encuentren en el interior del mapa.*

AccionMenu



Permite definir las opciones que aparecen en el menú de acciones del jugador. Cada entidad sólo permite definir una acción del menú.

Menú de acciones

Parámetros de esta entidad:

Activa
Con valor 'True' esta acción se aplicará al menú de acciones.
Con valor 'False' esta acción no se aplicará al menú de acciones.

Num_acción
Número de acción (entre 1 y 8).
Es el número de acción, de las 8 disponibles, que queremos cambiar. Las opciones están numeradas del 1 al 8 de arriba a abajo:

Verbo
Nombre de la acción.
Texto que representará la acción que llevará a acabo el jugador cuando la seleccione. Este texto puede ser cualquier cosa, aunque normalmente son verbos como: Leer, Comer, Usar, Robar, etc... o lo que se te ocurra.

Comentarios sobre esta entidad:
Esta lista son los números de acción correspondientes al orden de las acciones con sus respectivos nombres por defecto:

1 Usar
2 Entregar
3 Tomar
4 Mostrar
5 Comprar
6 Vender
7 Leer
8 Poner

Debes usar entidades '**AccionMenu**' para cambiar las opciones que te interese modificar para adaptar este menú a las necesidades de tu juego.
Recuerda que cada entidad es sólo para modificar el nombre de una acción del menú.
Sitúa estas entidades en el primer nivel de tu juego para que se vean en todos los niveles.
Para que este menú le aparezca al jugador al pulsar la tecla de inventario (tecla 'i') debes poner el valor 'Z' del parámetro '**Acciones_XY**' de la entidad Marcador con valor 1.

¿ Para que sirven estas acciones ?:
De hecho el efecto de estas acciones lo definirás tú con comandos en ficheros de comandos .INI. Básicamente se usan para procesar una determinada serie de comandos en función de la acción que seleccione el jugador combinada con el objeto que tenga seleccionado en su inventario de objetos.

¿ Cómo realiza el jugador una acción de este tipo ?:
1.- El jugador se sitúa en el lugar del nivel que desea realizar la acción.
2.- Hace aparecer el inventario de objetos con la tecla 'i'.
3.- Usando las teclas de selección de objeto del inventario (por defecto las flechas derecha e izquierda del teclado numérico), seleccionará el objeto que quiere que intervenga en la acción.
4.- Seguidamente usando las teclas de selección de acción del menú de acciones (por defecto las teclas arriba y abajo del teclado numérico), seleccionará la acción que quiere realizar combinada con el objeto que tiene seleccionado.
5.- Finalmente pulsará la tecla 'Acción' (por defecto la tecla de espacio) para realizar la acción combinada. El jugador verá en la parte inferior de la pantalla la acción combinada que intenta realizar, independientemente de si esa acción será procesada o no.

Yo como desarrolador del juego ... ¿ Cómo detecto y proceso esas acciones que realiza el jugador ?:
Cada vez que el jugador realiza una acción de este tipo, de forma completamente automática, Entidad 3D guarda en la variable '**ACCION_SELECCIONADA**' el texto de la acción que ha seleccionado el jugador, también guarda en la variable '**OBJETO_SELECCIONADO**' el número de objeto que tiene seleccionado en el inventario de objetos.
Pues ya te imaginas lo simple que resultará detectar la acción y el objeto que está utilizando el jugador cuando haya pulsado la tecla 'Acción'. Bastará usar en un fichero .INI asociado a una entidad, un simple 'IF COND' con dos condiciones para detectarlo.
Ejemplo 1:

```
IF COND ACCION_SELECCIONADA=Usar COND OBJETO_SELECCIONADO=3
ACCI TEXT El jugador quiere usar el objeto 3 aqui
ENDIF
```

Como ves, de ti depende realizar los comandos que creas convenientes para las combinaciones de acciones que establezcas en tu juego.
Observa que la detección del objeto usando la variable 'OBJETO_SELECCIONADO' no se realiza como en otros comandos de esta forma '3_tarjeta.act=1', sino que se debe utilizar el número de objeto directamente.
Si usas la detección de acciones en ficheros .INI asociados a entidades, es importante para que funcione, que el parámetro '**Boton_accion**' de la entidad que uses, esté a 'True'.

Si, eso está muy bien para detectar y procesar acciones en lugares determinados, pero ... ¿ como hago para por ejemplo detectar cuando el jugador se tome una pócima en cualquier momento y lugar del juego ?:
Para eso debes crear un fichero de comandos .INI independiente que se procesará cada vez que el jugador realice una acción de este tipo. El nombre de este fichero debes indicarlo en la entidad Configuracion parámetro '**FicheroAcciones**'.
Será pues en ese fichero de acciones donde deberemos detectar y procesar las posibles acciones que quiera realizar el jugador en cualquier momento y lugar del juego:
Ejemplo 2:
En un fichero que llamaremos '**accion_general.ini**' tendríamos lo siguiente:

```
IF COND ACCION_SELECCIONADA=Tomar COND OBJETO_SELECCIONADO=4
ACCI TEXT Te has tomado la pocima
ACCI QINV 4_pocima.act
ACCI DSAL 50
ENDIF
```

En el parámetro '**FicheroAcciones**' de la entidad '**Configuracion**' habremos puesto: **accion_general.ini**

También se pueden detectar acciones que prevemos que puede intentar el jugador y que no tienen una finalidad concreta en el juego pero no queremos que el jugador se quede sin algún tipo de respuesta:
Ejemplo 3:

```
IF COND ACCION_SELECCIONADA=Entregar COND OBJETO_SELECCIONADO=5
ACCI TEXT Ese libro de codigos es como chino para mi, quita ese libro de mi vista
ENDIF
```

El jugador también puede seleccionar alguna acción del menú de acciones y no seleccionar ningún objeto en el inventario de objetos, por lo que la acción será simplemente el propio nombre de la acción, sin objeto asociado. En este caso el contenido de la variable 'OBJETO_SELECCIONADO' valdrá 0.
Esto también lo podremos detectar de la siguiente forma:
Ejemplo 4:

```
IF COND ACCION_SELECCIONADA=Activar COND OBJETO_SELECCIONADO=0
ACCI TEXT El panel de control que has activado muestra el código: AF1D75
ENDIF
```

Otros ejemplos:
Ejemplo 5:
Vender unos planos (objeto número 16):

```
IF COND ACCION_SELECCIONADA=Vender COND OBJETO_SELECCIONADO=16
ACCI TEXT Gracias por los planos secretos. Toma tus 6000 euros
ACCI QINV 16_planos.act
ACCI OPE+ DINERO 6000
ENDIF
```

Ejemplo 6:
El jugador entrega un lingote de oro en espera de favores. Un favor puede ser hacer desaparecer un matón que está en una puerta que impide pasar al jugador:

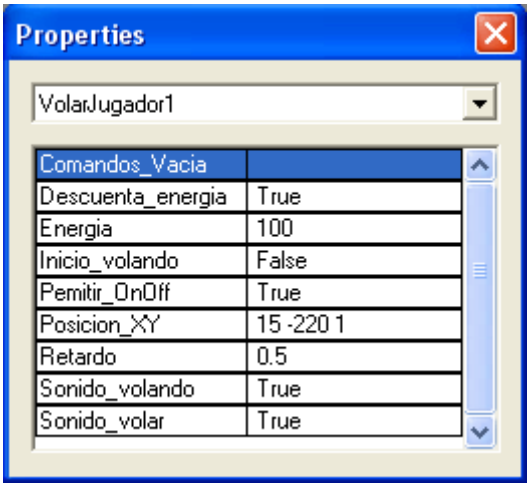
```
IF COND ACCION_SELECCIONADA=Entregar COND OBJETO_SELECCIONADO=17
ACCI TEXT Gracias por tu aportacion. ¡John, dejalo pasar!
ACCI QINVUSO 17_lingotes.act
ACCI VIS0 maton1.act 0
ENDIF
```

El funcionamiento de la tecla '**Acción**' cuando el jugador no tenga en pantalla el menú de acciones, sigue siendo igual que en anteriores versiones de Entidad 3D que no poseían el menú de acciones, es decir, lanzará el fichero de comandos asociado a la entidad si esa entidad tiene a 'True' el parámetro '**Boton_accion**'.

---

*» Coloca, en el primer nivel, tantas entidades '**AccionMenu**' como nombres de acciones quieras cambiar. Las entidades pueden estar situadas en cualquier parte del nivel mientras se encuentren en el interior del mapa.*
*» Si no quieres cambiar ninguna de las acciones que ya existen por defecto en el menú de acciones, no es necesario que uses ninguna entidad '**AccionMenu**' en todo tu juego.*
*» Consulta la sección Anexo Inventario para conocer más detalles sobre el **menú de acciones***

VolarJugador



Esta entidad permite configurar diversos parámetros para el modo volar del jugador.

Barra indicadora de energía de vuelo

Parámetros de esta entidad:

Pemitir_OnOff
Con 'True' el jugador podrá activar y desactivar el modo volar usando la tecla '**F**'.
Con 'False' el jugador no podrá controlar cuando puede volar.

Inicio_volando
Con valor 'True' el jugador comenzará el nivel volando.
Con valor 'False' el jugador comenzará el nivel andando.

Energia
Permite indicar la cantidad de energía inicial de la barra de energía de vuelo (entre 0 y 100).
Este valor podremos modificarlo durante el transcurso del juego usando el comando 'ENERGIAVOLAR'.

Posicion_XY
Permite indicar en que posición de la pantalla aparecerá la barra indicadora de energía de vuelo.
Aunque en este parámetro aparecen tres cajas para rellenar (X Y y Z), sólo debes usar X Y.
Un valor **'X' positivo** indicará que la coordenada de **X** se contará desde el margen izquierdo de la pantalla.
Un valor **'X' negativo** indicará que la coordenada de **X** se contará desde el margen derecho de la pantalla.
Un valor **'Y' positivo** indicará que la coordenada de **Y** se contará desde la parte superior de la pantalla.
Un valor **'Y' negativo** indicará que la coordenada de **Y** se contará desde la parte inferior de la pantalla.

Descuenta_energia
Descuenta energía al volar.
Con valor 'True' la barra de energía de vuelo se vaciará mientras el jugador esté volando.
Con valor 'False' la barra de energía de vuelo no vaciará mientras el jugador esté volando.

Retardo
Permite controlar la velocidad de vaciado de la barra de energía de vuelo.
El valor de este parámetro ha de ser un tiempo expresado en segundos.

Sonido_volar
Sonido de propulsión al volar el jugador.
Con valor 'True' se reproducirá un sonido cada vez que el jugador avance o retroceda cuando esté volando.
Con valor 'False' este sonido no se reproducirá.
El fichero de sonido de este efecto está en: **wav\volar.wav**

Sonido_volando
Sonido mientras vuela el jugador.
Con valor 'True' se reproducirá un sonido constante de fondo mientras el jugador esté volando.
Con valor 'False' este sonido no se reproducirá.
El fichero de sonido de este efecto está en: **wav\volando.wav**

Comandos_Vacia
Nombre del fichero de comandos .INI que se procesará cuando la barra de energía de vuelo se vacíe por completo.
Deja en blanco este parámetro si quieres que no se procese ningún fichero de comandos.

Las imágenes que forman la barra indicadora de energía de vuelo son:
bmp \ barras \ nivel_volar1.bmp
bmp \ barras \ nivel_volar2.bmp
bmp \ barras \ a_nivel_volar1.bmp
bmp \ barras \ a_nivel_volar2.bmp

COMANDOS DISPONIBLES

Comando SONF

Este comando se utiliza para interpretar un fichero de audio para crear un fondo musical continuo durante el juego.
SONF soporta ficheros de audio tipo: mp3, wav, mod, s3m, xm, it, mtm, mo3, mp1, mp2 y ogg.

Sintaxis: **ACCI SONF fichero_audio**
'fichero_audio': Es el nombre del fichero de audio a interpretar pero sin la extensión. El fichero ha de estar en la carpeta 'Wav' de 'Entidad 3D'.
Si el juego ya comenzó interpretando un fondo musical, este se detendrá y comenzará el nuevo fondo musical que hemos establecido con el comando SONF.
Ejemplo 1: Interpretar el fichero de audio musica.mod:
ACCI SONF musica
Ejemplo 2: Interpretar el fichero de audio ambiente.s3m:
ACCI SONF ambiente

Comando SONP

Este comando se utiliza para realizar una pausa en la interpretación del fondo musical activado con SONF o el fondo musical con el que comenzó el juego.

Sintaxis: **ACCI SONP**
Ejemplo: Hacer una pausa en el fondo musical del juego:
ACCI SONP

Comando SONR

Este comando se utiliza para reanudar la pausa de la música de fondo que hayamos detenido con el comando SONP.

Sintaxis: **ACCI SONR**
Ejemplo: Reanudar la música de fondo:
ACCI SONR

Comando SONI

Este comando se utiliza para interpretar un fichero de audio para un efecto de sonido concreto y puntual en el juego.
SONI soporta ficheros de audio tipo: mp3, wav, mod, s3m, xm, it, mtm, mo3, mp1, mp2 y ogg.

Sintaxis: **ACCI SONI fichero_audio**
'fichero_audio': Es el nombre del fichero de audio a interpretar pero sin la extensión. El fichero ha de estar en la carpeta 'Wav' de 'Entidad 3D'.
Ejemplo 1: Interpretar el fichero de audio bomba.mp3:
ACCI SONI bomba
Ejemplo 2: Interpretar el fichero de audio efecto.wav:
ACCI SONI efecto

Comando SON0

Detiene la interpretación de sonidos 3D que previamente se activaron con el comando 'SON1' o que ya estaban activos al dar comienzo el nivel.

Sintaxis del comando:
ACCI SON0 número_de_sonido3d
'número_de_sonido3d': Número de sonido 3D que se desea detener (entre 1 y 255).
Este número es el que asignamos en la casilla 'Num_sonido3d' de la entidad 'ElectricBolt' cuando creamos este sonido 3D.

Ejemplo: Queremos detener el sonido 3D número 7 que lo activamos con el comando 'ACCI SON1 7':
ACCI SON0 7

Consulta la sección del comando 'SON1' para saber como activar la interpretación de sonidos 3D.

Comando SON1

Reanuda la interpretación de sonidos 3D que se detuvieron previamente con el comando 'SON0' o que no estaban activos al dar comienzo el nivel.

Sintaxis del comando:
ACCI SON1 número_de_sonido3d
'número_de_sonido3d': Número de sonido 3D que se desea activar (entre 1 y 255).
Este número es el que asignamos en la casilla 'Num_sonido3d' de la entidad 'ElectricBolt' cuando creamos este sonido 3D.

Ejemplo: Queremos reanudar el sonido 3D número 10 que lo detuvimos con el comando 'ACCI SON0 10':
ACCI SON1 10

Consulta la sección del comando 'SON0' para saber como detener la interpretación de sonidos 3D.

Comandos SODF, SODP, y SODR

El comando 'SODF' se utiliza para interpretar un fichero de audio para crear un efecto se sonido continuo de fondo durante el juego.
'SODF' soporta ficheros de audio tipo: mp3, wav, mod, s3m, xm, it, mtm, mo3, mp1, mp2 y ogg.
Los comandos 'SODP' y 'SODR' nos permitirán detener y reanudar el sonido.

Sintaxis: **ACCI SODF fichero_audio volumen**
'fichero_audio': Es el nombre del fichero de audio a interpretar pero sin la extensión. El fichero ha de estar en la carpeta 'WAV' de 'Entidad 3D'.

'volumen': Es el nivel de volumen que se le aplicará al sonido. Admite valores de 0 a 100.

Este comando hace exactamente lo mismo que el comando 'SONF', pero mientras que este último está reservado para la música de fondo del juego, el comando 'SODF' sirve para reproducir sonidos de ambientes, efectos continuos pero sólo en zonas concretas, alarmas continuas, etc..., pero sin que la música de fondo deje de sonar, por lo tanto podremos tener al mismo tiempo una música de fondo, un sonido de ambiente de fondo y si además queremos hacer sonar un efecto de sonido puntual siempre podremos usar el comando 'SONI'.
'SODF' reproduce un sonido que el jugador no oirá como posicional, es decir, mientras el sonido esté activado el jugador lo oirá con el mismo volumen y sin variaciones en los canales de sonido derecho o izquierdo aunque se mueva por el nivel.
Si lo que quieres es usar sonidos posicionales, dispones de la entidad 'ElectricBolt'.

Ejemplo: Interpretar el fichero de audio **labofond.mp3** cuando el jugador entre en un laboratorio químico. El sonido produce efectos de burbujas y sonidos electrónicos mientras el jugador permanezca en el laboratorio.
En la puerta del laboratorio, en el lado del interior del laboratorio, colocar una entidad 'Command' con un fichero de comandos .INI asociado que contenga lo siguiente:
ACCI SODF labofond 75
Para detener el sonido cuando el jugador salga del laboratorio, colocaremos otra entidad 'Command' en la puerta, pero esta vez en la parte exterior del laboratorio, con un fichero de comandos .INI asociado que contenga lo siguiente:
ACCI SODP

---

*» También puedes usar el comando 'SODR' para situaciones en las que sólo utilices 'SODF' para cargar y comenzar a interpretar un sonido de fondo y posteriormente quieras activarlo y desactivarlo según diversas situaciones del juego. Para usar este comando simplemente tienes que poner **ACCI SODR**, reanudará el sonido que se haya detenido con 'SODP'.*

Comando TEXT

Presenta un texto de color azul y por unos segundos en la parte superior de la pantalla.

Sintaxis: **ACCI TEXT texto**
'texto': Es el texto que deseamos presentar. Si se presentan varios textos seguidos (con varios comandos ACCI TEXT), los textos se irán desplazando hacía arriba.
El texto puede contener mayúsculas, minúsculas, **Ñ**, **Ç** y diversos símbolos, aunque no soporta acentos ni símbolos **¡** o **¿**.
La cantidad de caracteres que podamos presentar por línea dependerá de la resolución de pantalla que se esté utilizando.
Si en alguna ocasión no aparece ni un solo carácter es que tenemos demasiados caracteres por línea.
Ejemplo: Presentar en pantalla el texto 'No puedes pasar por aqui':
ACCI TEXT No puedes pasar por aqui

Este comando también soporta la presentación de contenidos de variables, por lo que si por ejemplo llevamos un contador interno del dinero del jugador (pongamos que la variable se llame **DINERO**), para mostrárselo al jugador sólo tendríamos que usar la siguiente línea:
ACCI TEXT Llevas %DINERO% monedas de oro
Como ves si quieres presentar el contenido de una variable usando el comando TEXT tendrás que rodear esa variable con símbolos **%**.
En una misma línea puedes presentar la cantidad de variables que quieras, ya sean variables que contengan números o textos.

La forma de presentar contenidos de variables de esta manera, nos permite mostrar al jugador, de un modo muy sencillo, diversos datos que varían durante el juego como podrían ser por ejemplo: un nivel de energía, un nivel de poder mágico, fuerza, tarjetas de acceso, bolas mágicas que le faltan por recoger, etc...

» Recuerda que para presentar mensajes o contenidos de variables de una manera más atractiva y elaborada, dispones del comando 'MSJ1'.

Comando DIAL

Presenta un bloque de texto en pantalla.
Se puede utilizar para presentar lo que dicen los personajes o para presentar cualquier tipo de texto que requiere varias lineas.

Sintaxis: **ACCI DIAL fichero_dlg**
'fichero_dlg': Nombre del fichero .dlg que contiene las lineas de texto a presentar en pantalla.
Este fichero ha de estar en la carpeta 'Comandos' de 'Entidad 3D'.
El texto puede contener Ñ o Ç, aunque no soporta otros caracteres españoles.
La cantidad de caracteres que podamos presentar por linea dependerá de la resolución de pantalla que se esté utilizando.
Si en alguna ocasión no aparece ni un solo caracter es que tenemos demasiados caracteres por linea.
Las lineas se irán presentando en pantalla de cuatro en cuatro a intervalos de tiempo.
Ejemplo: El jugador colisiona con un personaje y este le dice algo:

Esto lo habremos conseguido con el comando:
ACCI DIAL personaje1.dlg
El diálogo lo habremos escrito en el fichero 'personaje1.dlg' que habremos copiado en la carpeta 'Comandos'.
Estos ficheros .dlg los podemos crear y editar con cualquier editor de textos, creando todas las líneas de texto que queramos que aparezcan en pantalla.

» Consulta la sección de la entidad 'SpecificActor' para ver ejemplos de como usar este comando conjuntamente con la entidad 'SpecificActor'.

Comando QINV

Se utiliza para eliminar un objeto en concreto del inventario de objetos del jugador.

Sintaxis: **ACCI QINV fichero_act_del_objeto**
'fichero_act_del_objeto': Es el nombre del fichero .act que representa este objeto.
Si el jugador no lleva el objeto que se desea eliminar, este comando no realizará nada.
Ejemplo: Bajo cierta circunstancia decidimos eliminar el objeto '2_espada.act' del inventario del jugador:
ACCI QINV 2_espada.act

Si el jugador tiene un objeto en su inventario y aún le queda más de un uso para ese objeto, el comando '**QINV**' seguirá funcionando de la misma forma, hará desaparecer el objeto del inventario del jugador, tenga los usos que tenga disponibles.

» Consulta los comandos 'QINVVENDER', 'QTIN' y 'QINVUSO', que también se utilizan para eliminar objetos del inventario del jugador.
» Consulta la sección 'Objetos' para ver ejemplos de esta entidad.

Comando QTIN

Se utiliza para eliminar todos los objetos de un tipo en concreto del inventario del jugador.

Sintaxis: **ACCI QTIN fichero_act_del_objeto**
'fichero_act_del_objeto': Es el nombre del fichero .act que representa este objeto.
Si el jugador no lleva ningún objeto del tipo que se desea eliminar, este comando no realizará nada.
Ejemplo: Queremos eliminar todos los objetos '3_bola.act' que pueda llevar el jugador en su inventario:
ACCI QTIN 3_bola.act

» Consulta los comandos 'QINVVENDER', 'QINV' y 'QINVUSO', que también se utilizan para eliminar objetos del inventario del jugador.
» Consulta la sección 'Objetos' para ver ejemplos de esta entidad.

Comando PINV

Se utiliza para añadir un objeto al inventario del jugador.
Normalmente el jugador va llenando su inventario de objetos con los objetos que va recogiendo por el escenario, este proceso de añadir estos objetos al inventario lo realiza automáticamente 'Entidad 3D', pero si queremos darle al jugador algún objeto mediante ciertas condiciones utilizando ficheros de comandos, pues utilizaremos este comando en el interior de algún fichero de comandos .ini asociando a alguna entidad.

Sintaxis: **ACCI PINV fichero_act_del_objeto usos nombre**
'fichero_act_del_objeto': Es el nombre del fichero .act que representa este objeto.
Si al jugador ya no le caben más objetos en el inventario, porque ya lleva 9 objetos, este comando no realizará nada, aunque advertirá al jugador de ello.
'usos': Entre 1 y 99
Número máximo de usos que podrá realizar el jugador con este objeto.
'nombre': Nombre del objeto
Nombre que tendrá el objeto.

Ejemplo 1: Añadir el objeto '7_libro.act' al inventario del jugador:
ACCI PINV 7_libro.act 1 Libro
Ejemplo 2: Añadir el objeto '8_oro.act' al inventario del jugador, pero si no se puede porque el inventario está lleno, presentar un mensaje:
IF COND inventario<20
ACCI PINV 8_oro.act 1 Oro
ELSE
ACCI TEXT No puedes llevar mas objetos, antes debes utilizar la bateria
ENDIF
Aquí utilizamos una variable de entorno llamada 'inventario' que actualiza 'Entidad 3D' de forma automática y que siempre contiene la cantidad de objetos que lleva el jugador en su inventario.
Ejemplo 3: Añadir el objeto '9_pocima.act' al inventario del jugador, pero que sólo pueda 'beber' 5 veces de la pócima:
ACCI PINV 9_pocima.act 5 Pocima

---

*» Puedes usar el comando 'PINV' con objetos que su representación 3D en forma de fichero .ACT ni siquiera exista, mientras los tengas representados en forma de iconos en el fichero **PrimerNivel_objetos_distintos.bmp**.*
» Consulta la sección de 'Objetos' para saber como hacer que cada objeto tenga una representación en forma de icono en el inventario de objetos del jugador.

Comando QINVUSO

Se utiliza para restar un uso de utilización de un objeto en concreto del inventario de objetos del jugador.

Sintaxis: **ACCI QINVUSO fichero_act_del_objeto**
'fichero_act_del_objeto': Es el nombre del fichero .act que representa este objeto.
Si el jugador no lleva el objeto, este comando no realizará nada.
Si después de reducir el número de usos, el objeto se queda sin usos al ser el último, automáticamente este objeto se eliminará del inventario de objetos del jugador.
Ejemplo 1:
Si el jugador usa la arma de energía 3 (que tiene en su inventario), produciremos el efecto de esa arma en su posición y reducimos una unidad el uso de esa arma (objeto '14_arma3.act'):
IF COND ACCION_SELECCIONADA=Usar COND OBJETO_SELECCIONADO=14
ACCI QINVUSO 14_arma3.act
ACCI ARMAENERGIA 3
ENDIF

Ejemplo 2:
Si el jugador se toma una pastilla (objeto 15_pastillas.act), le aumentaremos la capacidad de salto y reducimos una unidad el uso de esas pastillas, ya que se supone que es un frasco con varias pastillas:
IF COND ACCION_SELECCIONADA=Tomar COND OBJETO_SELECCIONADO=15
ACCI QINVUSO 15_pastillas.act
ACCI SALTO 950 30
ENDIF

Para que los ejemplos anteriores funcionen en cualquier lugar y momento del juego, esas líneas de comandos tendríamos que incluirlas en el fichero de comandos .INI que deberíamos indicar en el parámetro '**FicheroAcciones**' de la entidad 'Configuracion'.

» Consulta los comandos 'QINVVENDER', 'QTIN' y 'QINVUSO', que también se utilizan para eliminar objetos del inventario del jugador.
» Consulta la entidad 'ItemObject' para saber como asignar número de usos a los objetos.

Comando QINVVENDER

Se utiliza para realizar la venta de un objeto del inventario de objetos del jugador.

Sintaxis: **ACCI QINVVENDER**
Este comando lo único que hace es eliminar el objeto que tiene seleccionado el jugador en su inventario de objetos y aumentar la cantidad de dinero del jugador según el precio del objeto que se acaba de eliminar del inventario.
En definitiva es la venta de un objeto por parte del jugador.
Ejemplo: Un aldeano compra botas (objeto 12) pero no compra zapatos (objeto 11):
IF COND ACCION_SELECCIONADA=Vender COND OBJETO_SELECCIONADO=11
ACCI TEXT No compro zapatos, solo botas
ENDIF
IF COND ACCION_SELECCIONADA=Vender COND OBJETO_SELECCIONADO=12
ACCI TEXT Gracias por las botas, aqui tienes tu dinero
ACCI QINVVENDER
ENDIF
Si el jugador quiere vender las botas, seleccionará ese objeto en su inventario (objeto número 12) y seleccionará la acción '**Vender**' en el menú de acciones, seguidamente pulsará la tecla '**Acción**'. En este caso se cumpliría el segundo 'IF' del ejemplo.

Como ves, la simplicidad es máxima a la hora de vender algo por parte del jugador, pudiendo nosotros controlar quien acepta las ventas, que aceptan, cuando, etc.... todo dependerá del fichero de comandos .INI que realicemos y a que entidad lo asociemos.
En el ejemplo anterior, el aldeano podría estár representado por un fichero .ACT en una entidad 'SpecificActor' y a esta le asociaríamos un fichero de comandos .INI con los comandos del ejemplo anterior.
El comando '**QINVVENDER**' se encarga de eliminar el objeto que tenga el jugador seleccionado y aumentar el dinero del jugador con el valor que hayamos puesto en el parámetro '**Precio_venta**' de la entidad '**ItemObject**' de este objeto.

» *Consulta la entidad 'ItemObject' para saber como asignar precios a los objetos.*

Comando PONE

Este comando asigna un valor a una variable.

Sintaxis 1: **ACCI PONE nombre_variable=valor**
Sintaxis 2: **ACCI PONE nombre_variable=%nombre_variable%**
'nombre_variable': Nombre de la variable a la que queremos asignarle el valor.
Si la variable no existe, se creará con el valor asignado.
'valor': Valor que queremos asignarle a la variable. Pueden ser números, letras o símbolos.
'%nombre_variable%': Nombre de la variable cuyo contenido se copiará en la variable indicada en 'nombre_variable'.
Ejemplo 1: Cargar el valor 1 a la variable 'activado':
ACCI PONE activado=1
Ejemplo 2: Cargar el valor 56 a la variable 'monedas':
ACCI PONE monedas=56
Ejemplo 3: Cargar el valor 'mesa' a la variable 'objeto':
ACCI PONE objeto=mesa
Ejemplo 4: Cargar el valor 'la casa' a la variable 'sitio':
ACCI PONE sitio=la casa
Ejemplo 5: Copiar el valor que contiene la variable 'llaves' en la variable 'masllaves':
ACCI PONE masllaves=%llaves%

Comando PONA

Este comando asigna un valor al azar a una variable.

Sintaxis 1: **ACCI PONA nombre_variable=valor_max**
Sintaxis 2: **ACCI PONA nombre_variable=%nombre_variable%**
'nombre_variable': Nombre de la variable a la que queremos asignarle el valor al azar.
Si la variable no existe, se creará con el valor al azar.
'valor_max': Valor máximo que queremos que pueda tener el valor generado al azar. El comando generará un valor al azar entre 1 y 'valor_max'.
'%nombre_variable%': Nombre de una variable cuyo contenido se utilizará como el máximo valor que se generará.
Ejemplo 1: Generar un número al azar entre 1 y 5, el número generado dejarlo en la variable 'puerta'. Esta variable se podría utilizar para abrir una puerta entre 5 posibles:
ACCI PONA puerta=5
Ejemplo 2: Generar un número al azar entre 1 y lo que contenga la variable 'caminos', el resultado dejarlo en la variable 'opcion':
ACCI PONA opcion=%caminos%
Si la variable 'caminos' contiene el número 8, el número generado al azar será un número entre 1 y 8.
Si la variable 'caminos' contiene el número 5, el número generado al azar será un número entre 1 y 5.

Comando OPE+

Se utiliza para sumar un valor al contenido de una variable.

Sintaxis 1: **ACCI OPE+ nombre_variable valor**
Sintaxis 2: **ACCI OPE+ nombre_variable %variable%**
'nombre_variable': Nombre de la variable a la que le queremos sumar un valor.
'valor': Valor a sumar a la variable indicada en 'nombre_variable'.
'%variable%': Variable cuyo contenido será el que se sumará a la variable indicada en 'nombre_variable'.
Ejemplo 1: Sumar 2 a la variable 'energia'. Si antes del comando la variable 'energia' valía 6, después del siguiente comando valdrá 8:
ACCI OPE+ energia 2

Ejemplo 2: Sumar el contenido de la variable 'monedas'(que vale 3) al contenido de la variable 'dinero'(que vale 60) y el resultado dejarlo en la variable 'dinero':
ACCI PONE dinero=60
ACCI PONE monedas=3
ACCI OPE+ dinero %monedas%
El resultado de este ejemplo es que la variable 'dinero' finalmente valdrá 63

Comando OPE-

Se utiliza para restar un valor al contenido de una variable.

Sintaxis 1: **ACCI OPE- nombre_variable valor**
Sintaxis 2: **ACCI OPE- nombre_variable %variable%**
'nombre_variable': Nombre de la variable a la que le queremos restar un valor.
'valor': Valor a restar a la variable indicada en 'nombre_variable'.
'%variable%': Variable cuyo contenido será el que se restará a la variable indicada en 'nombre_variable'.
Ejemplo 1: Quitar 1 a la variable 'veces'. Si antes del comando la variable valía 10, después del siguiente comando valdrá 9:
ACCI OPE- veces 1

Ejemplo 2: Restar el contenido de la variable 'monedas'(que vale 7) a la variable 'dinero'(que vale 55) y el resultado dejarlo en la variable 'dinero':
ACCI PONE dinero=55
ACCI PONE monedas=7
ACCI OPE- dinero %monedas%
El resultado de este ejemplo es que la variable 'dinero' finalmente valdrá 48

Comando OPE*

Se utiliza para realizar operaciones de multiplicación con variables.

Sintaxis 1: **ACCI OPE* nombre_variable valor**
Sintaxis 2: **ACCI OPE* nombre_variable %variable%**
'nombre_variable': Nombre de la variable a la que le queremos multiplicar por un valor.
'valor': Valor a multimplicar a la variable indicada en 'nombre_variable'.
'%variable%': Variable cuyo contenido será el que se multiplicará a la variable indicada en 'nombre_variable'.
Ejemplo 1: Multiplicar 2 al contenido de la variable 'fuerza'. Si antes del comando la variable 'fuerza' valía 10, después del siguiente comando valdrá 20:
ACCI OPE* fuerza 2

Ejemplo 2: Multiplicar el contenido de la variable 'mana'(que vale 5) al contenido de la variable 'cantidad'(que vale 3) y el resultado dejarlo en la variable 'mana':
ACCI PONE mana=5
ACCI PONE cantidad=3
ACCI OPE* mana %cantidad%
El resultado de este ejemplo es que la variable 'mana' finalmente valdrá 15

Comando OPE/

Se utiliza para realizar operaciones de división con variables.

Sintaxis 1: **ACCI OPE/ nombre_variable valor**
Sintaxis 2: **ACCI OPE/ nombre_variable %variable%**
'nombre_variable': Nombre de la variable cuyo valor queremos dividir por otro valor.
'valor': Valor por el que se dividirá el contenido de la variable indicada en 'nombre_variable'.
'%variable%': Variable cuyo contenido será el que se utilice para dividir el contenido de variable indicada en 'nombre_variable'.

Ejemplo 1: Dividir el contenido de la variable 'peso' entre 2, el resultado dejarlo en la variable 'peso'. Si antes del comando la variable 'peso' valía 22, después del siguiente comando valdrá 11:
ACCI OPE/ peso 2

Ejemplo 2: Dividir el contenido de la variable 'escudo'(que vale 50) entre el contenido de la variable 'indice'(que vale 5) y el resultado dejarlo en la variable 'escudo':
ACCI PONE indice=5
ACCI PONE escudo=50
ACCI OPE/ escudo %indice%
El resultado de este ejemplo es que la variable 'escudo' finalmente valdrá 10

Ejemplo 3: Calcular el **40%** del valor de la variable **fuerza** y el resultado multiplicarlo por el contenido de la variable **incre** más **5**, el resultado final dejarlo en la variable **indice**. En el proceso no alterar en ningún momento las variables **fuerza** ni **incre**.
ACCI PONE fuerza=87
ACCI PONE incre=4
ACCI PONE temporal=%incre%
ACCI PONE indice=%fuerza%
ACCI OPE/ indice 100
ACCI OPE* indice 40
ACCI OPE+ temporal 5
ACCI OPE* indice %temporal%
El resultado de este ejemplo es que la variable 'indice' finalmente valdrá 306
Para un cálculo como el del ejemplo anterior es necesario realizar las operaciones comando a comando, en ningún caso Entidad 3D aceptará instrucciones como estas:
ACCI PONE indice=(%fuerza%/100)*40)*%incre%+5
ACCI PONE temporal=%incre%+5

Comando OPEC

Se utiliza para añadir textos al contenido de una variable.

Sintaxis 1: **ACCI OPEC nombre_variable texto**
Sintaxis 2: **ACCI OPEC nombre_variable %variable%**
'nombre_variable': Nombre de la variable a la que queremos añadir el texto.
'texto': Texto que se añadirá al contenido de la variable indicada en 'nombre_variable'. En realidad 'texto' puede ser letras, números o símbolos.
'%variable%': Variable cuyo contenido será el que se añadirá al contenido de la variable indicada en 'nombre_variable'.
Ejemplo 1: Añadir la letra '**A**' al contenido de la variable 'clave'. Si antes del comando la variable 'clave' contenía el texto '**PIEDR**', después del siguiente comando contendrá '**PIEDRA**:
ACCI OPEC clave A

Ejemplo 2: Añadir el texto '_amanece_mas_temprano' al contenido de la variable 'clave'. Si antes del comando la variable 'clave' contenía el texto 'No_por_mucho_madrugar', después del siguiente comando contendrá '**No_por_mucho_madrugar_amanece_mas_temprano**':
ACCI OPEC clave _amanece_mas_temprano
En el ejemplo anterior en lugar de separar las palabras por el símbolo '_' se pueden separar por el espacio normal, aunque para mostrar los ejemplos más claros he utilizado '_'.

Ejemplo 3: Añadir el contenido de la variable 'secuencia' al contenido de la variable 'clave' y dejar el resultado en la variable 'clave'.
Si antes del comando la variable 'clave' contenía el texto 'agencia', después del siguiente comando contendrá '**agencia374**':
ACCI PONE secuencia=374
ACCI PONE clave=agencia
ACCI OPEC clave %secuencia%

Comandos IF ELSE ENDIF

Se utilizan para procesar comandos bajo ciertas condiciones que podamos establecer durante el transcurso del juego.

Sintaxis:
IF COND condicion_o_condiciones
comandos_a_procesar_SI
ELSE
comandos_a_procesar_NO
ENDIF
'condicion_o_condiciones': Condición o condiciones que se han de cumplir.
'comandos_a_procesar_SI': Líneas de comandos que se procesarán si la condición se ha cumplido.
'comandos_a_procesar_NO': Líneas de comandos que se procesarán si la condición no se ha cumplido.

Ejemplo 1: Presentar un mensaje y activar un sonido si la variable 'acceso' vale 0, de lo contrario presentar un diálogo:
IF COND acceso=0

```
ACCI TEXT Acceso denegado
ACCI SONI efecto
ELSE
ACCI DIAL permitido.dlg
ENDIF
```

Ejemplo 2: Hacer aparecer un personaje en el caso de que la variable 'fase' valga 2 y el jugador lleve en su inventario un objeto '10_daga.act':

```
IF COND fase=2 COND 10_daga.act=1
ACCI VIS1 vendedor.act
ENDIF
```

Fíjate que si el jugador lleva más de un objeto '10_daga.act' la condición no funcionará ya que estamos preguntando si el jugador lleva 1 solo objeto de este tipo.

Ejemplo 3: Dar por finalizado el objetivo número 3 de la misión si el jugador alcanza la posición donde hemos situado una entidad 'Command', ha rescatado a un prisionero, lleva en su inventario de objetos la llave para salir (objeto por ejemplo '6_llave.act') y ha recogido los 10 cristales de energía:

```
IF COND objetivo3=0 COND rescatado=1 COND 6_llave.act=1 COND cristales=%totalcristales%
ACCI OBJ1 3
ACCI TEXT Objetivo número 3 cumplido
ACCI PONE objetivo3=1
ENDIF
```

Para que funcione con 10 cristales la variable **totalcristales** ha de contener el valor **10**.
Como ves este ejemplo, puedes usar variables en las condiciones para comprobar valores que puedan contener esas variables, e incluso comparar el contenido de una variable con el contenido de otra variable, en este caso el nombre de la variable que pongas a la derecha del símbolo '=' debe estar rodeada de símbolos **%**, ya que no son lo mismo estas dos condiciones:

```
IF COND cristales=%totalcristales%
IF COND cristales=totalcristales
```

Mientras la primera compara si el contenido de la variable '**cristales**' es igual al contenido de la variable 'totalcristales', la segunda compara si el contenido de la variable '**cristales**' es el texto **totalcristales**.

Ejemplo 4: Quitar una vida al jugador si una clave de acceso que tiene que introducir anteriormente (por ejemplo usando el comando 'INPU') no es la correcta. La palabra correcta es '**FOXTROT**':

```
IF COND clave_acceso!FOXTROT
ACCI TEXT Un rayo te matado, has perdido una vida
ACCI QVID
ELSE
ACCI TEXT Clave correcta
ENDIF
```

Como ves el símbolo **!** sirve para lo contrario que el símbolo **=**, en el ejemplo anterior la condición se da cumplida si el contenido de la variable 'clave_acceso' es distinto del texto 'FOXTROT'.
Los únicos símbolos que puedes utilizar en comparaciones son **=**, **!**, **<** y **>**, que en términos informáticos significan respectivamente '**igual a**', '**distinto de**', '**menor que**' y '**mayor que**'.

Ejemplo 5: Hacer desaparecer todas las piedras mágicas que el jugador lleve en su inventario de objetos si al menos lleva 3 de estas piedras. Si lleva ese mínimo de piedras, presentarle un mensaje informándole que ya tiene acceso a una habitación y modificar alguna variable que usarás en una condición para abrir la puerta de la habitación en otro punto del mapa.
Si el jugador anteriormente no ha activado un resorte, no comprobar nada de lo de las piedras y avisarle de lo que pasa.

```
IF COND toco_resorte=1
IF COND accesoroja=0
IF COND 7_piedra.act>2
ACCI QTIN 7_piedra.act
ACCI PONE accesoroja=1
TEXT Ahora tienes acceso a la habitacion roja
ELSE
TEXT No tienes la tres piedras mágicas
ENDIF
ELSE
TEXT Dirigete a la habitacon roja
ENDIF
ELSE
ACCI TEXT No has tocado el resorte del trono
ENDIF
```

Ejemplo 6: Tienes tres variables (var1, var2 y var3) que controlan tres eventos distintos del juego, las tres tienen el valor inicial de 0 y cuando cualquier evento se ha completado se pone a 1 la variable correspondiente.
Comprobar si alguno de los eventos se ha completado, si es así presentar una textura animada en el escenario que muestre al jugador un sencillo video del siguiente objetivo que debe cumplir.

```
ACCI PONE eventos=%var1%
ACCI OPE+ eventos=%var2%
```

```
ACCI OPE+ eventos=%var3%
IF COND eventos>0
ACCI TEA1 video
ELSE
ACCI TEXT Completa alguno de los tres eventos
ENDIF
```

Habilidad mental:
Tengo que reconocer que para una persona que nunca ha utilizado variables ni la lógica de las condiciones, aprender este tipo de sistema es un poco difícil, pero si quieres dotar a tu juego de una lógica de funcionamiento particular y tan compleja como quieras, es del todo imprescindible dominar mínimamente la utilización de estos comandos y condiciones que deberás introducir en ficheros con extensión .INI y asociar esos ficheros a distintas entidades que soportan este tipo de ficheros. En este manual, en el apartado de cada entidad, se indica si esa entidad soporta ficheros de comandos .INI y donde debes indicar el nombre del fichero para que Entidad 3D lo procese durante el juego.

En cualquier sistema de creación de juegos que te permita cierta libertad a la hora de establecer la trama de tu juego y que no te limite sólo a modificar cuatro datos introducidos en unas bonitas cajas, deslizar vistosos controles para variar otros tantos datos, o pulsar con el cómodo ratón sobre maravillosos y visuales botones, te encontrarás con algún sistema de 'scripts' o algo similar a los comandos de Entidad 3D.
Te comento esto por si este tipo de uso de comandos te echa hacia atrás a la hora de plantearte realizar un juego con algún sistema de creación de juegos.
Si bien usando el propio Entidad 3D o algún otro 'Game Maker' efectivamente te permite realizar juegos sin usar prácticamente ningún sistema de comandos, pronto observarás que si quieres dotar a tu juego de un mecanismo particular siempre tendrás que enfrentarte, por muy simple o complejo que sea, a algún tipo de sistema similar a la lógica de la programación de computadoras.
Si observas detenidamente los ejemplos anteriores observarás que no es tan difícil y a lo poco que te vayas dando cuenta de las combinaciones que puedes realizar con variables, comandos y condiciones verás la potencia que tienes en tus manos para controlar lo que sucede en tu juego.
Es verdad que este sistema de codificar comandos es muy similar a como se realiza para un programa de computadora y siendo el lema de Entidad 3D es '**Creación de juegos 3D sin programar**' parece un poco engañoso, pero ni de lejos es tan complejo como programar en cualquier lenguaje de programación, incluidos los lenguajes 'script', tampoco será tan potente como esos lenguajes pero para tener un mínimo control sobre lo que hace el jugador y lo que pasa en tu juego, este lenguaje de comandos de Entidad 3D es más que suficiente, en la mayoría de los casos, para por lo menos encontrar una manera de realizar lo que tienes en mente usando de forma creativa combinaciones de diversos comandos y entidades.

Para encontrar los comandos correctos, usados con las variables correctas, las condiciones correctas, en el lugar correcto y en el momento correcto, es necesario cierta habilidad mental y capacidad de abstracción del mecanismo de funcionamiento que tienes en mente y que deberás traducir en comandos de Entidad 3D que en muchos casos no hacen exactamente lo que quieres, y en ocasiones encontrar una manera equivalente de hacerlo puede ser todo un desafío.
Esa habilidad mental de la que hablo te aseguro que no se nace con ella, pero tampoco se aprende en dos días, se adquiere con la práctica, viendo ejemplos de otras personas observando como han resuelto la situación que querían realizar, entendiendo muy bien que es lo que hace cada comando, asumiendo las limitaciones de Entidad 3D, teniendo mucho interés y sobre todo dedicándole mucho mucho mucho tiempo.

---

*» Es bastante recomendable que te acostumbres a realizar este tipo de escalonamientos de líneas que ves en los ejemplos cuando uses comandos '**IF COND ELSE ENDIF**' para que te resulte más fácil de entender que se procesa al cumplir la condición, que se procesa si no se cumple o que condición está dentro de otra.*
» Detrás de los 'IF' puedes poner tantas condiciones 'COND' como quieras, pero para que se procesen los comandos debajo del 'IF' tendrán que cumplirse todas las condiciones.
*» Recuerda que los ficheros .INI debes ponerlos siempre en la carpeta '**comandos**' de Entidad 3D y que son simples ficheros de texto que puedes crear con cualquier editor normal de textos tipo Notepad o similar.*
» Consulta la sección 'Abriendo puertas' para ver ejemplos de como condicionar aperturas de puertas usando condiciones y comandos.

Comando EXIT

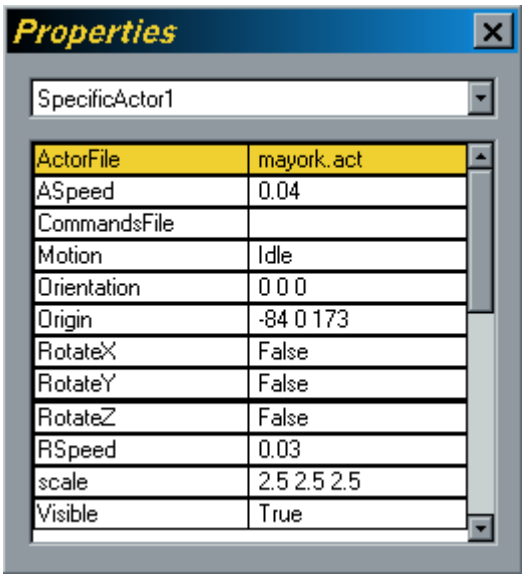
Este comando deja de procesar cualquier comando que le siga, dando por finalizado el proceso de comandos del fichero .INI que se esté procesando.

Sintaxis: **ACCI EXIT**

Ejemplo: Incrementar el nivel de salud del jugador en función de la cantidad de pastillas que haya recogido. Informarle si no ha conseguido salud al no disponer de pastillas:

```
IF COND pastillas>0 COND pastillas<4
ACCI TEXT Consigues 10 puntos de salud
ACCI DSAL 10
ACCI PONE pastillas=0
ACCI EXIT
ENDIF
IF COND pastillas>3
ACCI TEXT Consigues 30 puntos de salud
ACCI DSAL 30
ACCI PONE pastillas=0
ACCI EXIT
ENDIF
ACCI TEXT No tienes pastillas
```

Comando VIS0

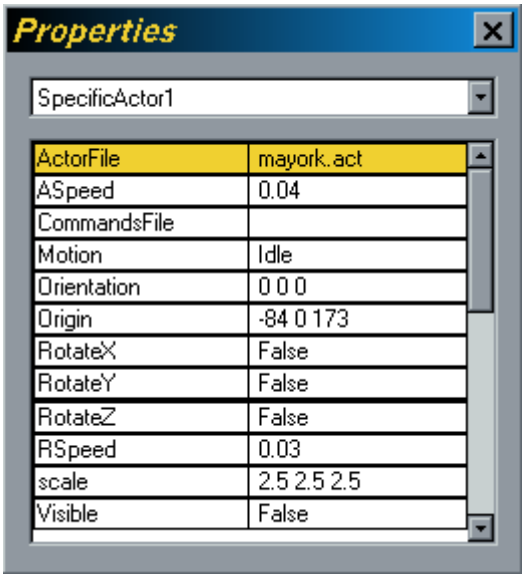


Hace invisible un personaje secundario creado en el escenario con la entidad 'SpecificActor' y con el parámetro 'Visible' de esta entidad puesta como 'True'.

Sintaxis: **ACCI VIS0 nombre_act_personaje tiempo**
'nombre_act_personaje': Nombre del fichero .act que representa ese personaje en el escenario.
'tiempo': Tiempo, en segundos, que queremos que permanezca invisible el personaje.
Usar 0 si se quiere hacer invisible indefinidamente.
Ejemplo: En nuestro escenario hemos colocado un personaje secundario con la entidad 'SpecificActor' de esta forma:

Este personaje será visible desde el comienzo del juego ya que 'Visible' está a 'True', pero durante el transcurso del mismo podremos hacerlo desaparecer con el siguiente comando:
ACCI VIS0 mayork.act 0
Si queremos que sólo permanezca 6 segundos invisible pondremos lo siguiente:
ACCI VIS0 mayork.act 6

Comando VIS1

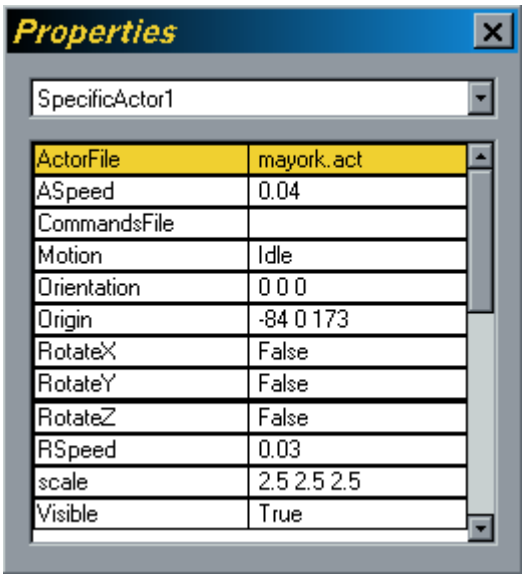


Hace visible un personaje secundario creado en el escenario con la entidad 'SpecificActor' y con el parámetro 'Visible' de esta entidad puesta como 'False'.

Sintaxis: **ACCI VIS1 nombre_act_personaje tiempo**
'nombre_act_personaje': Nombre del fichero .act que representa ese personaje en el escenario.
'tiempo': Tiempo, en segundos, que queremos que permanezca visible el personaje.
Usar 0 si se quiere hacer visible indefinidamente.
Ejemplo: En nuestro escenario hemos colocado un personaje secundario con la entidad 'SpecificActor' de esta forma:

Este personaje será invisible desde el comienzo del juego ya que 'Visible' está a 'False', pero durante el transcurso del mismo podremos hacerlo aparecer con el siguiente comando:
ACCI VIS1 mayork.act 0
Si queremos que sólo permanezca 15 segundos visible pondremos lo siguiente:
ACCI VIS1 mayork.act 15

Comando MOTC



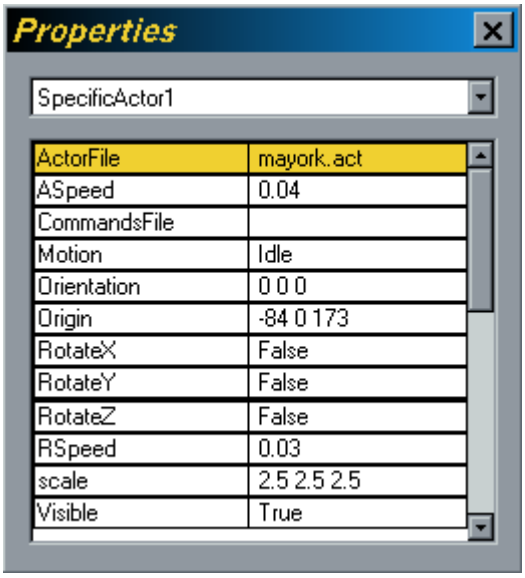
Cambia de animación un personaje secundario creado en el escenario con la entidad 'SpecificActor'.

Sintaxis: **ACCI MOTC nombre_act_personaje nombre_animacion**
'nombre_act_personaje': Nombre del fichero .act que representa ese personaje en el escenario.
'nombre_animacion': Nombre de la nueva animación que le queremos aplicar al personaje. Esta animación ha de ser una de las disponibles en el fichero .act de este personaje.
Ejemplo: En nuestro escenario hemos colocado un personaje secundario con la entidad 'SpecificActor' de esta forma:

Este personaje aparecerá en el escenario realizando la animación 'Idle' tal y como indica el parámetro 'Motion' de esta entidad, pero durante el transcurso del juego podremos cambiar esta animación con el siguiente comando:
ACCI MOTC mayork.act Entregar
En este caso le hemos aplicado la animación 'Entregar' que este personaje realizará continuamente.

Comando MOTI

Cambia de animación un personaje secundario creado en el escenario con la entidad 'SpecificActor', pero a diferencia del comando MOTC, MOTI sólo realizará una sola vez la animación designada en este comando, volviendo al instante a la anterior animación cíclica que estaba realizando el personaje.



Sintaxis: **ACCI MOTI nombre_act_personaje nombre_animacion**
'nombre_act_personaje': Nombre del fichero .act que representa ese personaje en el escenario.
'nombre_animacion': Nombre de la nueva animación que le queremos aplicar al personaje. Esta animación ha de ser una de las disponibles en el fichero .act de este personaje.
Ejemplo: En nuestro escenario hemos colocado un personaje secundario con la entidad 'SpecificActor' de esta forma:

Este personaje aparecerá en el escenario realizando la animación 'Idle' tal y como indica el parámetro 'Motion' de esta entidad, pero durante el transcurso del juego queremos que el personaje realice la animación de saltar por un instante. Lo realizaremos con el siguiente comando:
ACCI MOTI mayork.act JumpStart
En este caso le hemos aplicado la animación 'JumpStart' que este personaje realizará sólo por el tiempo que dure esa animación.

Comando PERS

Cambia el personaje protagonista durante el transcurso del juego.

Sintaxis: **ACCI PERS numero_de_personaje**
'numero_de_personaje': Con valor 1, el personaje protagonista será el que contiene el fichero '*PrimerNivel*_player1.act'. Con valor 2, el personaje protagonista será el que contiene el fichero '*PrimerNivel*_player2.act.
Estos dos ficheros han de estar en la carpeta 'Actors' de 'Entidad 3D'.
MUY IMPORTANTE: Si no quieres utilizar un segundo aspecto para tu personaje protagonista y no dispones de un '*PrimerNivel*_player2.act', copia también el '*PrimerNivel*_player1.act' como '*PrimerNivel*_player2.act', el caso es tener siempre estos dos ficheros en la carpeta 'Actors', de lo contrario el programa se interrumpirá de forma brusca.
Ejemplo 1: Tenemos a nuestro personaje protagonista en el fichero 'salon_player1.act':

Por lo que al dar comienzo el juego aparece así:

Por la razón que sea el personaje sufre una transformación y queremos que cambie de aspecto. Tenemos el fichero 'salon_player2.act' conteniendo este personaje:

Y durante el transcurso del juego cambiaremos al personaje protagonista a este aspecto ... :

... mediante este simple comando:
ACCI PERS 2
Para volverlo al aspecto inicial sólo tendremos que hacer:
ACCI PERS 1

Ejemplo 2: Queremos hacer que nuestro protagonista se transforme en un lobo cuando entre en una habitación por una puerta en concreto, y que cuando salga de esa habitación se vuelva a transformar en un hombre. El nombre del primer nivel de nuestro juego se llamará 'salon'.
Suponemos que tenemos los ficheros 'hombre.act' y 'lobo.act', los copiamos en la carpeta 'Actors' pero cambiándoles el nombre:
hombre.act --> salon_player1.act
lobo.act --> salon_player2.act
Ahora vamos a añadir a nuestro escenario una entidad 'Command' justo en la posición de la entrada de la puerta, editamos sus propiedades y en el parámetro 'CommandsFile' ponemos por ejemplo 'transforma.ini'. Luego creamos un fichero llamado de la misma forma, 'transforma.ini', en la carpeta 'comandos' y lo rellenamos con lo siguiente:

```
IF COND aspecto=0
ACCI PONE aspecto=1
ACCI PERS 2
ELSE
ACCI PONE aspecto=0
ACCI PERS 1
ENDIF
```

¿ Cómo funciona esto ?. La entidad 'Command' se utiliza cuando nos interesa procesar comandos cuando el jugador simplemente pase por la posición donde hemos colocado esta entidad. Pues bien, partimos de que la variable 'aspecto', que nos hemos inventado, vale 0 cuando comienza nuestro juego. La primera vez que el jugador pase por nuestro punto, por la entrada de la puerta, como la variable 'aspecto' vale 0 pues la condición se cumple y se procesan 'ACCI PONE aspecto=1' y 'ACCI PERS 2', con lo que nuestro personaje cambiará de aspecto debido al comando 'PERS 2', también ponemos la variable 'aspecto' a 1 para que la próxima vez que el jugador pase por ese punto la condición no se cumpla y de esa forma se procese lo que hay detrás de 'ELSE', 'ACCI PONE aspecto=0' y 'ACCI PERS 1'. Con 'PERS 1' hacemos que el protagonista vuelva a su aspecto anterior, mientras que con 'PONE aspecto=0' hacemos que el ciclo se vuelva a repetir indefinidamente.

Comando OPCI

Permite plantear opciones al jugador.
Con este comando podremos hacer que el jugador tenga que seleccionar una opción entre cuatro posibles para seguir jugando, además de poder procesar comandos según la opción elegida.
Mientras el jugador no seleccione alguna opción no podrá moverse.
El jugador podrá seleccionar una de las opciones que se planteen pulsando las teclas del '1' al '4'.

Sintaxis: **ACCI OPCI fichero_dlg fichero_ini_1 fichero_ini_2 fichero_ini_3 fichero_ini_4**
'fichero_dlg': Nombre del fichero de texto .dlg que contendrá un texto previo al planteamiento de las opciones, un diálogo de algún personaje, etc.., además de la relación de opciones posibles de que dispone el jugador.
'fichero_ini_1': Nombre del fichero .ini que se procesará en el caso de que el jugador seleccione la opción '1'.
'fichero_ini_2': Nombre del fichero .ini que se procesará en el caso de que el jugador seleccione la opción '2'.
'fichero_ini_3': Nombre del fichero .ini que se procesará en el caso de que el jugador seleccione la opción '3'.
'fichero_ini_4': Nombre del fichero .ini que se procesará en el caso de que el jugador seleccione la opción '4'.
En una línea que por ejemplo sólo queramos proponer 3 opciones al jugador, no es necesario que pongamos cuatro ficheros .ini, sino 3.
Ejemplo: Tenemos el fichero 'dialogo_ladron.dlg' (en la carpeta 'comandos) que contiene las siguientes líneas:

```
El ladron te dice: Para pasar por este puente debes pagar.
.
.
Que haces ?:
1.- Le das la bolsa de oro
2.- Le atacas con la espada
3.- Le pides por favor que te deje pasar
4.- No haces nada
```

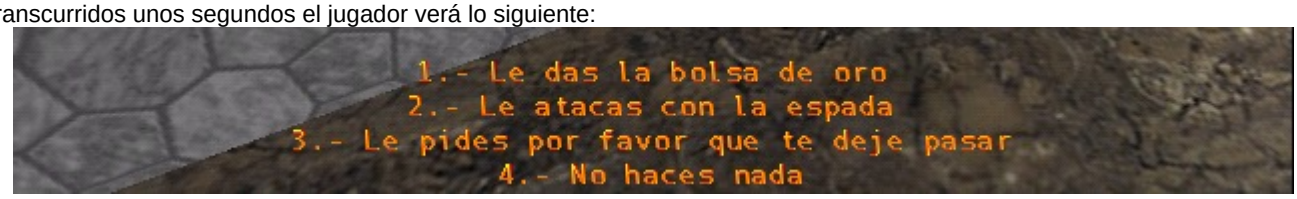
En un momento dado durante el transcurso de nuestro juego queremos plantear esas 4 opciones al jugador, por lo que utilizamos el comando 'OPCI' de la siguiente forma:

```
IF COND hablando_ladron=0
ACCI OPCI dialogo_ladron.dlg ladron_opc1.ini ladron_opc2.ini ladron_opc3.ini ladron_opc4.ini
ACCI PONE hablando_ladron=1
ENDIF
```

Evidentemente tendremos que tener los cuatro ficheros .ini (ladron_opc1.ini ladron_opc2.ini ladron_opc3.ini ladron_opc4.ini) en la carpeta 'comandos', que contendrán cada uno de ellos los comandos que queremos procesar para cada una de las opciones, según la que seleccione el jugador.
Pero vamos a ver que sucede cuando se procese la serie de comandos anteriores:

En primer lugar he utilizado la condición 'IF COND hablando_ladron=0' para que una vez utilizado el comando 'OPCI' que viene a continuación, no se vuelvan a plantear otra vez las mismas opciones al jugador, si no te podrías encontrar que una vez el jugador haya seleccionado una opción y como no se ha podido mover del sitio donde se ha lanzado este comando 'OPCI', pues vuelva a procesarse otra vez el comando 'OPCI' y así sucesivamente sin poder salir de este bucle. Por eso utilizo una variable que pongo a '1' con 'ACCI PONE hablando_ladron=1' para que no se vuelva a cumplir la condición y por lo tanto no se vuelva a procesar este 'OPCI'.
Si de todas formas quieres que este 'OPCI' pueda volver a funcionar repetidas veces durante el transcurso del juego, ya sabes lo que te toca hacer ... pues en algún momento del desarrollo del juego tendrás que procesar una línea como esta: 'ACCI PONE hablando_ladron=0'
Supongamos que ya se ha procesado nuestra línea con el 'OPCI', esto es lo primero que verá el jugador:

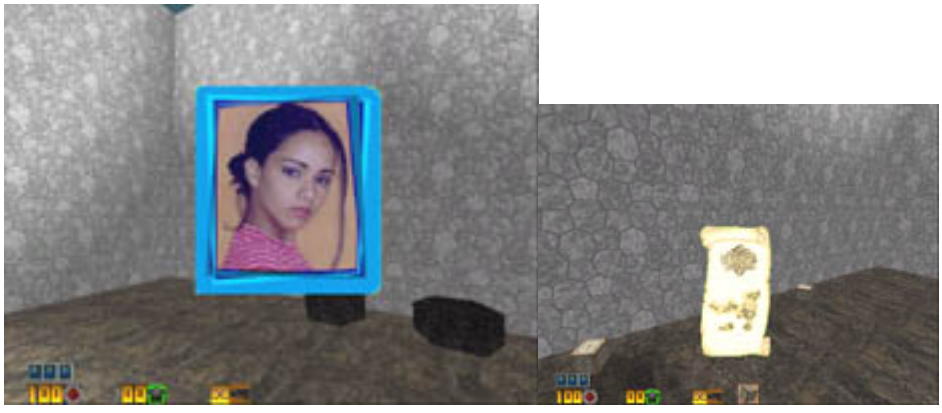
Inmediatamente el jugador no podrá moverse.
Transcurridos unos segundos el jugador verá lo siguiente:



El jugador seguirá sin poder moverse y viendo las opciones hasta que seleccione una de las opciones que le planteamos. Una vez el jugador haya pulsado la tecla del número de opción correspondiente, automáticamente se procesará el fichero .ini que le corresponde y finalmente el jugador podrá moverse.
Fíjate que en estos ficheros .ini que se procesan en función de la opción seleccionada por el jugador podríamos tener otro 'OPCI' con otras opciones y así sucesivamente, por lo que una ingeniosa utilización de estos comandos nos permitirá incluso crear laboriosos diálogos entre personajes del juego y el jugador.
Como puedes ver, el contenido del fichero de texto .dlg ha de ser de una manera concreta para que todo funcione bien y que al jugador se le presenten en pantalla todas las líneas de texto hasta las 4 líneas finales que quedarán congeladas hasta que el jugador seleccione una opción.
En el ejemplo anterior hemos utilizado las 4 primeras líneas para presentar mínimamente las opciones que vendrán a continuación, que corresponden a las 4 líneas siguientes, pero podríamos haber utilizado 8 líneas para la presentación o 12, etc...
Como ves, es recomendable utilizar grupos de 4 en 4 líneas para que al final se le presenten al jugador sólo las líneas que corresponden a las opciones a plantear, ya que son esas 4 lineas finales las que Entidad 3D mantiene en pantalla hasta que el jugador seleccione una opción, mientras que las anteriores las va mostrando en pantalla a intervalos regulares de tiempo.

Comando IMG1

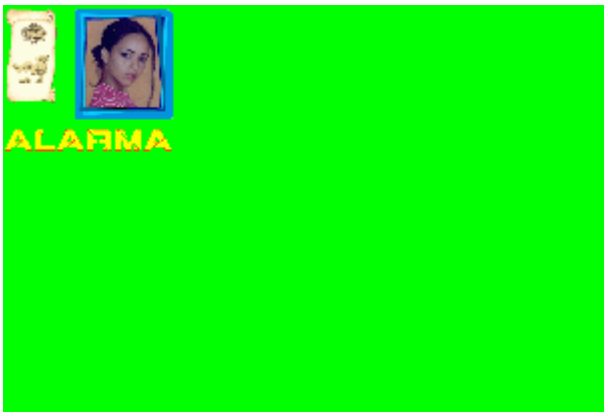
Permite superponer imágenes 2D fijas en pantalla, mientras el desarrollo del juego en 3D transcurre detrás de la imagen.



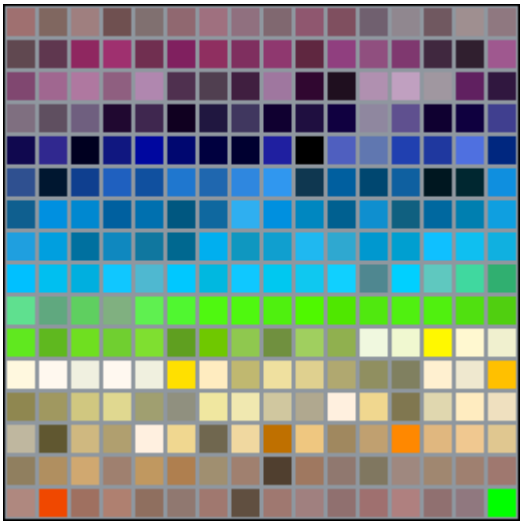
Las imágenes que queramos superponer en pantalla deben ser porciones de imagen que deben estar contenidas en una imagen en formato BMP con el siguiente formato de nombre de fichero:

*PrimerNivel*_graficos_superpuestos.bmp

Este fichero ha de estar en la carpeta 'bmp', ha de ser de una resolución de 1600x1200 y con una paleta gráfica de 256 colores (del 0 al 255) y se considerará el color 255 (el último) como el color transparente.
Un fichero de ejemplo de este tipo tendría este aspecto:



Esta es la paleta de colores de la imagen anterior:



Sintaxis del comando IMG1:
ACCI IMG1 número_de_imagen tiempo xo1 yo1 xo2 yo2 px py
'número_de_imagen': Número de imagen (entre 1 y 5).
Este número lo asignaremos manualmente al utilizar este comando para superponer una imagen en pantalla. Podemos utilizar hasta 5 imágenes simultáneas en pantalla.
Este número de imagen también será el que deberemos utilizar en el comando 'IMG0' para referirnos a una imagen concreta para hacerla desaparecer de la pantalla.
'tiempo': Tiempo en segundos que deseamos que permanezca la imagen en pantalla, transcurrido ese tiempo la imagen desaparecerá automáticamente.
Utiliza el valor 0 para no establecer ningún tiempo, de esta forma la imagen permanecerá indefinidamente, hasta que utilicemos el comando 'IMG0' para hacerla desaparecer.

'xo1' e 'yo1': Coordenadas X e Y dentro de la imagen '*PrimerNivel*_graficos_superpuestos.bmp', que establece la esquina superior izquierda de la porción de imagen que queremos superponer en pantalla.

'xo2' e 'yo2': Coordenadas X e Y dentro de la imagen '*PrimerNivel*_graficos_superpuestos.bmp', que establece la esquina inferior derecha de la porción de imagen que queremos superponer en pantalla.

'px' y 'py': Porcentaje de X e Y en la pantalla, que establecerá la posición donde se mostrará la imagen superpuesta.
Tenemos los valores del 1 a 100 para fijar la X (en horizontal) y los valores del 1 a 100 para fijar la Y (en vertical).
Si ponemos los valores 0 para 'px' y 0 para 'py' la imagen siempre aparecerá centrada en pantalla, independientemente de la resolución a la que se esté jugando.

Ejemplo: Tenemos el fichero 'puente_graficos_superpuestos.bmp' con varios gráficos sobre un fondo verde que nos hace de color transparente.
Queremos extraer el gráfico que pone 'Alarma' de nuestro fichero de gráficos superpuestos y ponerlo en la parte superior de la pantalla durante 8 segundos.
Primero determinamos las coordenadas X e Y de un rectángulo imaginario que envolvería el gráfico 'Alarma' dentro del BMP 'puente_graficos_superpuestos.bmp'.

Luego calculamos unos valores de X e Y para que el gráfico quede centrado en la parte superior de la pantalla. En este caso asignaremos 20 para el porcentaje horizontal y 6 para el porcentaje vertical.
Finalmente decidimos que designaremos nuestro gráfico como el número 5.
Con todos estos datos ya estamos en disposición de crear la línea de comandos que presentará la imagen 'Alarma' en pantalla. Esta es la línea:
IMG1 5 8 3 277 379 328 20 6
... y este sería el resultado en pantalla una vez procesada esa línea:

En este caso el gráfico 'Alarma' desaparecerá transcurridos 8 segundos.

Consulta la sección del comando 'IMG0' para ver como puedes hacer desaparecer un gráfico superpuesto que has puesto en pantalla con tiempo 0.

Comando IMG0

Se utiliza para hacer desaparecer imágenes superpuestas que previamente se hayan situado con el comando 'IMG1'.

Sintaxis del comando:
ACCI IMG0 número_de_imagen
'número_de_imagen': Número de imagen que se desea hacer desaparecer (entre 1 y 5).
Este número es el mismo que le asignamos a la imagen que hicimos aparecer con el comando 'IMG1'.

Ejemplo: Utilizando el comando IMG1 de esta forma:
IMG1 2 0 3 277 379 328 20 6
... hemos situado en pantalla las letras 'Alarma':

... y como hemos utilizado 0 para fijar el tiempo de permanencia en pantalla, las letras no desaparecerán hasta que utilicemos esta simple línea de comando:
IMG0 2

Consulta la sección del comando 'IMG1' para conocer la sintaxis exacta de este comando que sitúa imágenes superpuestas en pantalla.

Comandos:

OVERLAY1 y OVERLAY0

Permite superponer imágenes 2D fijas en pantalla, mientras el desarrollo del juego en 3D transcurre detrás de las imágenes superpuestas.
Este comando tiene un funcionamiento similar al comando 'IMG1', aunque con el comando 'OVERLAY1' podrás utilizar imágenes de 24bits de color y cargadas desde ficheros .BMP independientes.
Esto te permitirá personalizar tu juego definiendo todo tipo de marcadores, indicadores, mensajes gráficos, ventanas, cuadros o lo que quieras presentar en pantalla según las situaciones que se produzcan en tu juego.

Sintaxis del comando **OVERLAY1** para presentar imágenes:
ACCI OVERLAY1 número_de_imagen tiempo x y

Sintaxis del comando **OVERLAY0** para hacer desaparecer imágenes:
ACCI OVERLAY0 número_de_imagen

'número_de_imagen': Número de imagen (entre 1 y 64).
Este número lo asignaremos manualmente al utilizar este comando para superponer una imagen en pantalla. Podemos utilizar hasta 64 imágenes simultáneas en pantalla.
Este número de imagen también será el que deberemos utilizar en el comando '**OVERLAY0**' para referirnos a una imagen concreta para hacerla desaparecer de la pantalla.
'tiempo': Tiempo en segundos que deseamos que permanezca la imagen en pantalla, transcurrido ese tiempo la imagen desaparecerá automáticamente.
Utiliza el valor 0 para no establecer ningún tiempo, de esta forma la imagen permanecerá indefinidamente, hasta que utilicemos el comando '**OVERLAY0**' para hacerla desaparecer.
'x': Coordenada X para establecer, junto con la coordenada Y, la posición de la imagen en pantalla.
Un valor **'x' positivo** indicará que la coordenada de **x** se contará desde el margen izquierdo de la pantalla.
Un valor **'x' negativo** indicará que la coordenada de **x** se contará desde el margen derecho de la pantalla.
Un valor **0** centrará la imagen horizontalmente.
'y': Coordenada Y para establecer, junto con la coordenada X, la posición de la imagen en pantalla.
Un valor **'y' positivo** indicará que la coordenada de **y** se contará desde la parte superior de la pantalla.
Un valor **'y' negativo** indicará que la coordenada de **y** se contará desde la parte inferior de la pantalla.
Un valor **0** centrará la imagen verticalmente.

Cada imagen que queramos superponer en pantalla debe estar en un fichero .BMP con un nombre en concreto si se quiere usar para un juego en concreto. El formato del nombre de los ficheros debe ser:

*PrimerNivel*_overlay1.bmp ..... para la imagen número 1
*PrimerNivel*_overlay2.bmp ..... para la imagen número 2
*PrimerNivel*_overlay3.bmp ..... para la imagen número 3
.
.
... hasta un máximo de 64 imágenes

En el caso de usar imágenes de 24bits de color y queramos usar transparencias, deberá existir también una imagen 'Alpha' para la imagen correspondiente. El formato del nombre de estos ficheros 'Alpha' debe ser:

*PrimerNivel*_a_overlay1.bmp ..... para la imagen número 1
*PrimerNivel*_a_overlay2.bmp ..... para la imagen número 2
*PrimerNivel*_a_overlay3.bmp ..... para la imagen número 3

.
.
... hasta un máximo de 64 imágenes

Los colores que debes utilizar para crear un fichero con 'Alpha' deben ser únicamente el blanco y el negro, siendo las zonas coloreadas con negro las que se considerarán transparentes en su imagen correspondiente, mientras que las zonas con color blanco serán las que aparecerán en pantalla.
No se pueden usar niveles de grises para crear semi-transparencias, ya que el comando 'OVERLAY1' no soporta 'Alpha Blending'.
Tanto los ficheros .BMP de las imágenes como los que contienen la imagen en 'Alpha', han de estar en la carpeta '**bmp\overlays**' de Entidad 3D, y deben cumplir las siguientes condiciones:
Pueden ser de 256 colores (paleta de colores del 0 al 255) y se considerará el color 255 (el último de la paleta) como el color transparente. En este caso no se necesita ningún fichero 'Alpha' para la transparencia.
Pueden ser de 24bits de color, pero para crear transparencias tendremos que crear su fichero 'Alpha' correspondiente.
Las dimensiones de la imagen han de ser alguna de las siguientes: 16x16, 32x32, 64x64, 128x128, 256x256, 512x512, 1024x1024. En algunos casos otros tamaños también podrían funcionar según la tarjeta gráfica que uses, pero en ningún caso utilices tamaños con valores impares.

Ejemplo 1:
Situar una imagen de una fotografía de la cara de una persona, en la parte superior izquierda de la pantalla durante 5 segundos:
Nuestro juego lo componen tres niveles distintos llamados: parque, calle y plaza, aunque sea cual sea el nivel donde queramos situar la imagen, el nombre del fichero .BMP necesario, para por ejemplo usar la imagen número 4, tendría que llamarse:

*parque*_overlay4.bmp

parque_overlay4.bmp (256x256 24bits)

Esta sería la línea de comandos que situaría la imagen en pantalla:
ACCI OVERLAY1 4 5 10 18
Este ejemplo mostrará la imagen, en la coordenada situada **10** pixeles contados desde la izquierda de la pantalla y a **18** pixeles contados desde la parte superior de la pantalla. El **5** es para que el mensaje sólo permanezca 5 segundos en pantalla:

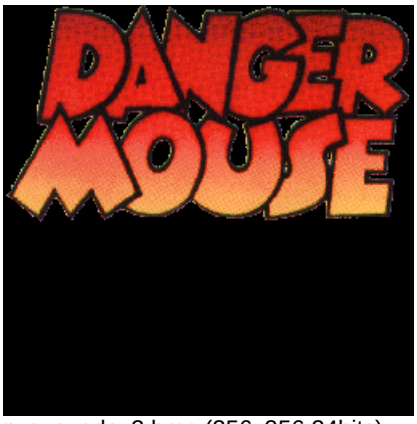
Resultado final

Ejemplo 2:
Situar un título centrado en la parte superior de la pantalla durante un tiempo indefinido. La imagen tiene zonas transparentes y tiene un tamaño de 251x135 que no se adapta a los tamaños permitidos por el comando 'OVERLAY1':
Siguiendo con los mismos nombres para los niveles del ejemplo anterior, los nombres de los ficheros .BMP necesarios, para por ejemplo usar la imagen número 2, tendrían que llamarse de la siguiente forma:

*parque*_overlay2.bmp
*parque*_a_overlay2.bmp

Imagen original (251x135 24bits)

Imagen con transparencia (251x135 24bits)



parque_overlay2.bmp (256x256 24bits)

parque_a_overlay2.bmp (256x256 24bits)

Esta sería la línea de comandos que situaría la imagen en pantalla:
ACCI OVERLAY1 2 0 0 5

Resultado final

... y esta línea de comandos para hacer desaparecer la imagen:
ACCI OVERLAY0 2

Como ves, todo el trabajo de edición de estos gráficos para prepararlos para Entidad 3D, no está explicado con detalle porque no es tarea de esta sección y varía de un editor gráfico a otro.

» Consulta el apartado titulado '**Coordenadas x y relativas a los bordes de pantalla**' de la sección del comando 'MSJ1' para saber como funciona el sistema de posicionamiento de elementos en pantalla usando coordenadas **x y** positivas y/o negativas.
» Recuerda que todos los ficheros .BMP que necesita el comando 'OVERLAY1' han de estar en la carpeta **'bmp\overlays'** de Entidad 3D.
» Nunca borres el fichero '**overlay_base.bmp**' que hay en la carpeta '**bmp\overlays**' de Entidad 3D.

Comandos:
MSJ1 y **MSJ0**

Permite superponer mensajes con textos, números o símbolos en cualquier lugar de la pantalla.

Sintaxis del comando **MSJ1** para presentar mensajes:
ACCI MSJ1 número_de_mensaje tiempo tipo_letra tamaño x y texto

Sintaxis del comando **MSJ0** para hacer desaparecer mensajes:
ACCI MSJ0 número_de_mensaje

'número_de_mensaje': Número de mensaje (entre 1 y 128).
Este número lo asignaremos manualmente al utilizar este comando para superponer cada texto, por lo que de esta forma podemos utilizar hasta 128 textos simultáneamente en pantalla.
Este número de mensaje es el que deberemos utilizar en el comando '**MSJ0**' para referirnos a un mensaje en concreto para hacerlo desaparecer de la pantalla.
'tiempo': Tiempo en segundos que deseamos que permanezca el mensaje en pantalla, transcurrido ese tiempo la imagen desaparecerá automáticamente.
Utiliza el valor 0 para no establecer ningún tiempo, de esta forma el mensaje permanecerá en pantalla indefinidamente, o hasta que utilicemos el comando '**MSJ0**' para hacerlo desaparecer.
'tipo_letra': Tipo de letra a utilizar para el mensaje. Consulta más abajo los tipos de letra y tamaños que hay disponibles.
'tamaño': Tamaño de letra a utilizar para el mensaje. Consulta más abajo los tipos de letra y tamaños que hay disponibles.
'x': Coordenada X para establecer, junto con la coordenada Y, la posición del mensaje en pantalla.

Un valor **'x' positivo** indicará que la coordenada de **x** se contará desde el margen izquierdo de la pantalla.

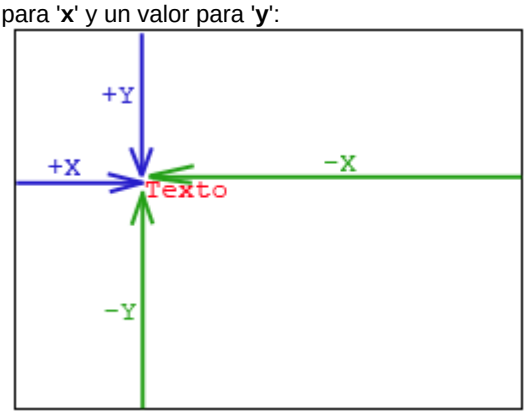
Un valor **'x' negativo** indicará que la coordenada de **x** se contará desde el margen derecho de la pantalla.
Un valor **0** centra el texto del mensaje horizontalmente.
'y': Coordenada Y para establecer, junto con la coordenada X, la posición del mensaje en pantalla.

Un valor **'y' positivo** indicará que la coordenada de **y** se contará desde la parte superior de la pantalla.

Un valor **'y' negativo** indicará que la coordenada de **y** se contará desde la parte inferior de la pantalla.
Un valor **0** centra el texto del mensaje verticalmente.
'texto': Texto que queremos hacer aparecer.
Si usas los tipos de letra 1 a 26 (tamaños 1 y 2) sólo se admiten letras mayúsculas sin acentos, también se admiten las mayoría de los símbolos, incluidos **¡ ¿** y las letras **Ç** y **Ñ**.
Usando los tipos de letra 1 a 3 (tamaño 3) tampoco se aceptan acentos, aunque si se aceptan letras minúsculas.
También se pueden indicar nombres de variables si los nombres los introduces entre símbolos **%**, para que se muestre el contenido de las variables.

Coordenadas **x y** relativas a los bordes de pantalla:
Las coordenadas **x y** que debemos indicar en este comando, y en otros comandos y entidades que usan este mismo tipo de posicionamiento, serán siempre unas coordenadas con respecto a los bordes de la pantalla, de esta forma nos permitirá situar mensajes de forma correcta, independientemente de la resolución de pantalla que use el jugador.
Con sólo dos datos, **x** e **y**, podremos situar cualquier mensaje en pantalla. Si usamos valores positivos la coordenada se contará desde la esquina superior izquierda de la pantalla, mientras que si usamos valores negativos se contará desde la

esquina inferior derecha de la pantalla. Podemos combinar positivos con negativos para situar el mensaje donde queramos, pero siempre sólo dos datos, un valor para '**x**' y un valor para '**y**':



Ejemplo 1: Queremos situar el texto **CONSEGUIDO** en la parte superior izquierda de la pantalla:
ACCI MSJ1 1 8 5 2 20 30 CONSEGUIDO

Este ejemplo muestra el texto **CONSEGUIDO** con el tipo de letra **5** y el tamaño de letra **2**, en la coordenada situada **20** pixeles contados desde la izquierda de la pantalla y a **30** pixeles contados desde la parte superior de la pantalla. El **8** es para que el mensaje sólo permanezca 8 segundos en pantalla:

Ejemplo 2: Situar, por un tiempo indefinido, el texto '**NIVEL:**' seguido del contenido de la variable **ENERGIA** en la parte superior derecha de la pantalla. Usar un tipo de letra pequeño:
ACCI MSJ1 2 0 3 1 -300 10 NIVEL:%ENERGIA%

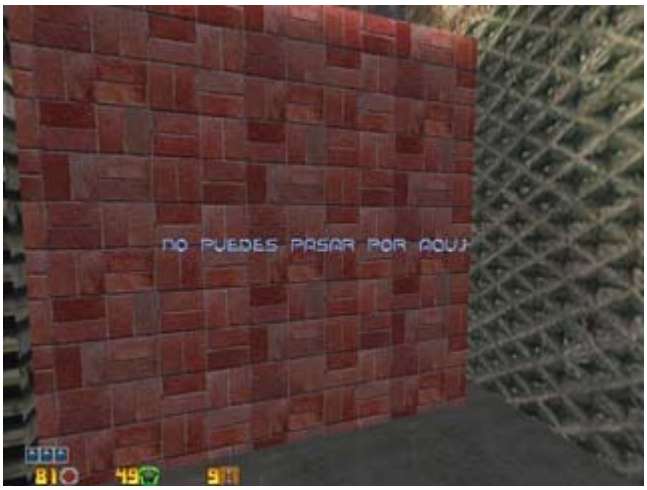
Si queremos eliminar este mensaje de la pantalla tendremos que usar el siguiente comando en algún punto de nuestro juego:
ACCI MSJ0 2

Ejemplo 3: Situar, por un tiempo indefinido, el contenido de la variable '**CAJAS**' en la parte inferior derecha de la pantalla:
ACCI MSJ1 3 0 14 2 -110 -45 %CAJAS%

Ejemplo 4: Situar, durante 10 segundos, el texto '**DINERO:**' seguido del contenido de la variable '**MONEDAS**' en la parte inferior izquierda de la pantalla. Se debe usar un tipo de letra para el texto y otro diferente para el valor de la variable:
ACCI MSJ1 4 10 20 1 15 -70 DINERO:
ACCI MSJ1 5 10 21 1 125 -70 %MONEDAS%

Ejemplo 5: Situar, durante 3 segundos, el texto '**ALERTA**' en la parte superior de la pantalla y centrado horizontalmente:
ACCI MSJ1 5 3 18 2 0 5 ALERTA

Ejemplo 6: Situar, durante 5 segundos, el texto '**NO PUEDES PASAR POR AQUI**' en el centro de la pantalla:
ACCI MSJ1 6 5 8 1 0 0 NO PUEDES PASAR POR AQUI



Tipos de letras y tamaños disponibles:

Tipo de letra:**1** Tamaño:**2**

Tipo de letra:**2** Tamaño:**2**

Tipo de letra:**3** Tamaño:**2**

Tipo de letra:**4** Tamaño:**2**

Tipo de letra:**5** Tamaño:**2**

Tipo de letra:**6** Tamaño:**2**

Tipo de letra:**7** Tamaño:**2**

Tipo de letra:**8** Tamaño:**2**

Tipo de letra:**9** Tamaño:**2**

Tipo de letra:**10** Tamaño:**2**

Tipo de letra:**11** Tamaño:**2**

Tipo de letra:**12** Tamaño:**2**

Tipo de letra:**13** Tamaño:**2**

Tipo de letra:**14** Tamaño:**2**

Tipo de letra:**15** Tamaño:**2**

Tipo de letra:**16** Tamaño:**2**

Tipo de letra:**17** Tamaño:**2**

Tipo de letra:**18** Tamaño:**2**

Tipo de letra:**19** Tamaño:**2**

Tipo de letra:**20** Tamaño:**2**

Tipo de letra:**21** Tamaño:**2**

Tipo de letra:**22** Tamaño:**2**

Tipo de letra:**23** Tamaño:**2**

Tipo de letra:**24** Tamaño:**2**

Tipo de letra:**25** Tamaño:**2**

Tipo de letra:**26** Tamaño:**2**

Tipo de letra:**1** Tamaño:**1**

Tipo de letra:**2** Tamaño:**1**

Tipo de letra:**3** Tamaño:**1**

Tipo de letra:**4** Tamaño:**1**

Tipo de letra:**5** Tamaño:**1**

Tipo de letra:**6** Tamaño:**1**

Tipo de letra:**7** Tamaño:**1**

Tipo de letra:**8** Tamaño:**1**

Tipo de letra:**9** Tamaño:**1**

Tipo de letra:**10** Tamaño:**1**

Tipo de letra:**11** Tamaño:**1**

Tipo de letra:**12** Tamaño:**1**

Tipo de letra:**13** Tamaño:**1**

Tipo de letra:**14** Tamaño:**1**

Tipo de letra:**15** Tamaño:**1**

Tipo de letra:**16** Tamaño:**1**

Tipo de letra:**17** Tamaño:**1**

Tipo de letra:**18** Tamaño:**1**

Tipo de letra:**19** Tamaño:**1**

Tipo de letra:**20** Tamaño:**1**

Tipo de letra:**21** Tamaño:**1**

Tipo de letra:**23** Tamaño:**1**

Tipo de letra:**23** Tamaño:**1**

Tipo de letra:**24** Tamaño:**1**

Tipo de letra:**25** Tamaño:**1**

Tipo de letra:**26** Tamaño:**1**

Tipo de letra:**1** Tamaño:**3**

Tipo de letra:**2** Tamaño:**3**

Tipo de letra:**3** Tamaño:**3**

Tipo de letra:**4** Tamaño:**3**

Tipo de letra:**5** Tamaño:**3**

Tipo de letra:**6** Tamaño:**3**

» Los mensajes que presentes con este comando permanecerán en pantalla aunque el jugador pase a otro nivel, de ti dependerá hacerlos desaparecer o no.
» Los mensajes donde presentes contenidos de variables, si esos contenidos de variables se modifican durante el juego, el mensaje que esté en pantalla con los contenidos anteriores no se actualizará, de ti depende volver a generar el comando 'MSJ1' correspondiente para que se actualice en pantalla, en ese caso utiliza el mismo número de mensaje tanto para situar el mensaje por primera vez como para actualizarlo.
*» Los ficheros .BMP que forman las letras los encontrarás en las carpetas: '**bmp\fuentes\1**', '**bmp\fuentes\2**' y '**bmp\consola\640x480**' de Entidad 3D.*

Comando TEM1

Activa temporizadores.
Se utiliza para que transcurrido un tiempo dado se procesen comandos.

Sintaxis del comando:
ACCI TEM1 número_de_temporizador tiempo nombre_fichero_ini
'número_de_temporizador': Número de temporizador que se desea activar (entre 1 y 15).
Este número lo asignaremos manualmente.
Podemos utilizar hasta 15 temporizadores simultáneamente.
Este número también será el que deberemos utilizar en el comando 'TEM0' para referirnos a un temporizador en concreto para desactivarlo.
'tiempo': Tiempo que ha de transcurrir para que se lance el procesamiento del 'fichero_ini' que se especifica a continuación.
'fichero_ini': Nombre del fichero .ini que contendrá los comandos a procesar una vez transcurrido el tiempo establecido. Este fichero .ini es un fichero normal como el resto de ficheros .ini que se utilizan en Entidad 3D. Puede contener todo tipo de comandos o secuencias de IF COND ENDIF. Como es habitual para estos ficheros, han de estar en la carpeta 'comandos'.

El comando TEM1 se puede utilizar en infinidad de situaciones. Por ejemplo, puede servir para hacer explotar bombas retardadas, presentar un mensaje al cabo de un tiempo, permitir abrir una puerta en un determinado tiempo, hacer aparecer o desaparecer personajes secundarios, etc... Todo dependerá de la combinación que realices con estos temporizadores y los comandos disponibles en Entidad 3D.

Ejemplo: Queremos crear una situación en la que el jugador, pulsando un botón, desbloquee una puerta que se encuentra a una cierta distancia, sólo permitiremos al jugador abrir esa puerta en un determinado tiempo, transcurrido ese tiempo la puerta ya no se podrá abrir, a menos que el jugador vuelva a repetir todo el proceso.
Evidentemente para montar toda esta situación intervienen diversos comandos y entidades, pero para el ejemplo que nos ocupa no me voy a centrar demasiado en los elementos que no sean del temporizador.

Lo que está claro es que tendremos que disponer de una puerta (entidad 'Door') donde le hayamos asociado un fichero .ini para condicionar su apertura. Ese fichero .ini le llamaremos por ejemplo 'puerta.ini', fichero que crearíamos en la carpeta 'comandos' con el siguiente contenido:

```
IF COND puerta_abrir=1
ACCI TEXT Abriendo puerta de seguridad
ENDIF
```

(Los comandos anteriores se procesarán cuando el jugador intente abrir la puerta).

Al lado de algo que represente el botón de desbloqueo situaremos una entidad 'Command' a la cual le asociaremos un fichero .ini que por ejemplo le llamaremos 'boton.ini', fichero que crearíamos en la carpeta 'comandos' con el siguiente contenido:

```
ACCI TEM1 3 10 bloquear_puerta.ini
ACCI TEXT Puerta de seguridad desbloqueada
ACCI PONE puerta_abrir=1
```

(Los comandos anteriores se procesarán cuando el jugador se acerque al botón).

Pues sólo nos queda crear el fichero 'bloquear_puerta.ini' que hemos puesto en el comando del temporizador, este fichero .ini contendría lo siguiente:

```
ACCI TEXT Puerta de seguridad bloqueada
ACCI PONE puerta_abrir=0
```

(Los comandos anteriores se procesarán transcurridos 10 segundos que comienzan a contar a partir de que el jugador 'toca' o se acerca al botón).

Fíjate que si el jugador alcanza la puerta en menos de 10 segundos la variable 'puerta_abrir' valdrá 1 porque ese es el valor que le hemos asignado en 'ACCI PONE puerta_abrir=1', por lo tanto la condición 'COND puerta_abrir=1' de la puerta se cumplirá y la puerta se abrirá.

Modificaciones posibles:
Un detalle a comentar es que si el jugador abre la puerta y la atraviesa, cuando la puerta se cierre ya no podrá volver a abrirla ya que de todas formas el temporizador cuando alcanzó los 10 segundos puso la variable 'puerta_abrir' a 0. Siempre hablando en el supuesto de que el jugador no tenga ninguna manera de volver a alcanzar el punto donde se encuentra el botón de desbloqueo.
Puede que te interese que el mecanismo funcione así, pero si quieres que el jugador pueda abrir la puerta desde cualquier lado una vez que consiga abrirla por primera vez, tendrías que variar el contenido del fichero 'puerta.ini' y dejarlo como sigue:

```
IF COND puerta_abrir=1
ACCI TEM0 3
ACCI TEXT Abriendo puerta de seguridad
ENDIF
```

(Con 'ACCI TEM0 3' detenemos el temporizador 3, el que estamos utilizando, si el jugador consigue abrir la puerta).
También tendremos que modificar el contenido del fichero 'boton.ini' para evitar que el jugador pueda volver a activar el temporizador. Todo el contenido de ese fichero .ini lo dejaríamos como sigue:

```
IF COND puerta_abrir=0
ACCI TEM1 3 10 bloquear_puerta.ini
ACCI TEXT Puerta de seguridad desbloqueada
ACCI PONE puerta_abrir=1
ENDIF
```

Consulta la sección del comando 'TEM0' para desactivar temporizadores que aún no hayan alcanzado su tiempo establecido.

Comando TEM0

Desactiva temporizadores.
Se utiliza para desactivar temporizadores que se activaron previamente con el comando 'TEM1' y que aún no hayan alcanzado su tiempo de retardo.

Sintaxis del comando:
ACCI TEM0 número_de_temporizador
'número_de_temporizador': Número de temporizador que se desea desactivar (entre 1 y 15).
Este número es el que asignamos en el comando TEM1 cuando activamos el temporizador.

Ejemplo: Queremos desactivar el temporizador número 2 que activamos previamente con la siguiente línea de comandos:

```
ACCI TEM1 2 30 explosion.ini
```

Para desactivarlo utilizaremos este simple comando:

```
ACCI TEM0 2
```

De esta forma si el temporizador 2 no ha alcanzado aún los 30 segundos, no seguirá contando y por lo tanto ya no se llegarán a procesar los comandos contenidos en el fichero 'explosion.ini'.

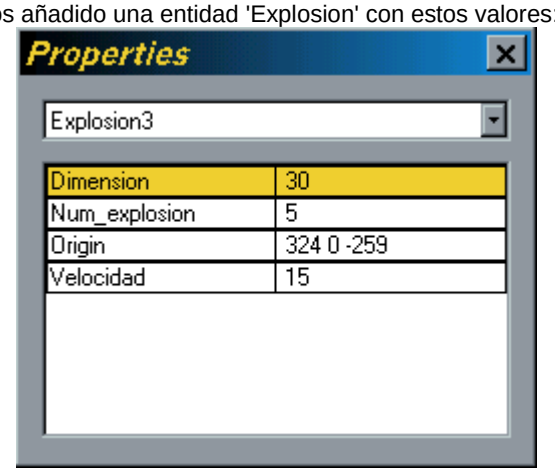
Consulta la sección del comando 'TEM1' para ver un ejemplo conjunto de comandos TEM1 y TEM0.

Comando EXPL

Este comando nos permite activar las explosiones (entidades 'Explosion') que tengamos esparcidas por el escenario.

Sintaxis: **ACCI EXPL número_de_explosión**
'número_de_explosión': Número de explosión que hemos asignado en la caja 'Num_explosion' de la entidad 'Explosion' que nos interesa activar el efecto de explosión.
Si tenemos diversas entidades 'Explosion' con el mismo número de explosión, con un solo comando 'EXPL' con ese número de explosión, se activarán simultáneamente todas las explosiones con ese número.

Ejemplo: En nuestro escenario hemos añadido una entidad 'Explosion' con estos valores:



Para poder activar esta explosión en un momento dado durante el transcurso del juego, tendremos que utilizar esta simple línea de comandos:
ACCI EXPL 5

Consulta la sección de la entidad 'Explosion' para ver las características de esta entidad que se utiliza junto con el comando 'EXPL'.

Comando BLQ0

Detiene la animación de bloques del escenario que se activaron previamente con el comando 'BLQ1' o que ya estaban animándose al dar comienzo el nivel.

Sintaxis del comando:
ACCI BLQ0 número_de_bloque
'número_de_bloque': Número de bloque que se desea detener su animación (entre 1 y 255).
Este número es el que asignamos en la casilla 'Num_bloque' de la entidad 'ModelController' cuando creamos este bloque animado.

Ejemplo: Queremos detener la animación del bloque número 3 que lo activamos con el comando 'ACCI BLQ1 3':
ACCI BLQ0 3

Consulta la sección del comando 'BLQ1' para saber como reanudar la animación de un bloque.

Comando BLQ1

Reanuda la animación de bloques del escenario que se detuvieron previamente con el comando 'BLQ0' o que no estaban animándose al dar comienzo el nivel.

Sintaxis del comando:
ACCI BLQ1 número_de_bloque
'número_de_bloque': Número de bloque que se desea animar (entre 1 y 255).
Este número es el que asignamos en la casilla 'Num_bloque' de la entidad 'ModelController' cuando creamos este bloque animado.

Ejemplo: Queremos reanudar la animación del bloque número 3 que lo detuvimos con el comando 'ACCI BLQ0 3':
ACCI BLQ1 3

Consulta la sección del comando 'BLQ0' para saber como detener la animación de un bloque.

Comando QVID

Quita una vida al jugador.
Este comando mata al jugador al instante, quitándole una de las vidas que lleve. Si era la última vida, evidentemente la partida habrá finalizado.

Sintaxis del comando:
ACCI QVID

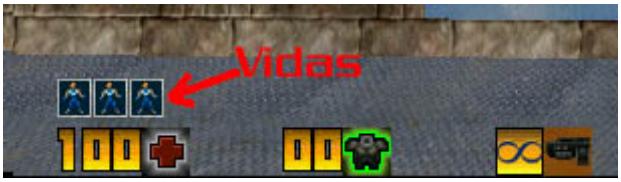
Este comando nos da la posibilidad de establecer la muerte del jugador en cualquier situación que se nos ocurra, no sólo a consecuencia de los disparos de los enemigos o al tocar la lava.
Algunos ejemplos de estas situaciones podrían ser:

- Al tocar o pisar alguna zona del escenario.
- Al hablar con un asesino.
- Al cruzar un campo de energía.
- Al elegir incorrectamente algún interruptor.
- Después de haber transcurrido un tiempo límite.
- Por falta de oxígeno en algún compartimento.
- Al tomar un veneno.

Ejemplo: Queremos hacer que el jugador pierda una vida:
ACCI QVID

Consulta la sección del comando 'DVID' que se utiliza para dar una vida al jugador.

Comando DVID



Da una vida al jugador.
Este comando da al instante una vida al jugador, produciendo al mismo tiempo un efecto de sonido que es el mismo que se produce al recoger una vida extra.

Sintaxis del comando:
ACCI DVID

Con este comando podemos premiar al jugador con una vida extra según la situación y condiciones que establezcamos en el desarrollo del juego.
No confundir este comando con la entidad 'ItemVida', las dos proporcionan una vida extra al jugador, pero mientras la entidad 'ItemVida' es un elemento que tenemos que situar en el escenario y la vida extra la conseguirá el jugador al recoger ese elemento, el comando 'DVID' lo debemos utilizar de la misma forma que utilizamos el resto de comandos de Entidad 3D, para establecer la lógica y desarrollo de nuestro juego.
Algunos ejemplos para utilizar este comando podrían ser:

- Al llegar al final de la fase.
- Una mago entrega al jugador energía vital.
- Al final del nivel el jugador consigue tantas vidas como secretos haya descubierto.
- Al combinar elementos del inventario del jugador en una pócima.
- Si el jugador mata a dos guardianes.
- El jugador ha completado unas tareas en un tiempo dado.
- Un personaje entrega una vida al jugador a cambio de unos objetos.

Ejemplo: Queremos premiar al jugador con una vida extra:
ACCI DVID

Consulta la sección del comando 'QVID' que se utiliza para quitar una vida al jugador.

Comando FINJ

Se utiliza para dar por finalizado el juego una vez el jugador ha completado todas las misiones y ha alcanzado su objetivo final.

Sintaxis del comando:
ACCI FINJ efecto
'efecto': Sin utilidad por el momento, con poner siempre 0 es suficiente.

La función de este comando es detener por completo el juego y presentar una pantalla de final de juego donde por ejemplo se puede felicitar al jugador o mostrarle el tesoro que ha encontrado o a la persona que ha encontrado, etc...
El jugador sólo le quedará la opción de hacer aparecer el menú con la tecla 'ESC' para decidir salir del juego o volver a jugar otra partida.
La imagen que queramos hacer aparecer ha de estar en la carpeta 'BMP' con este nombre:
*PrimerNivel*_final_juego.bmp

Para un juego que por ejemplo comience con un nivel llamado 'tunel.bsp' el nombre del fichero gráfico de final de juego podría ser 'tunel_final_juego.bmp'.

Ejemplo: Comando para dar por finalizado el juego y que presentará nuestra pantalla de final de juego:
ACCI FINJ 0
En realidad no sólo puedes utilizar ficheros gráficos .BMP para presentar la pantalla de final de juego, también puedes usar algún formato gráfico de los siguientes:
BMP, JPG, GIF, LBM, IFF, PBM, PCX, PGM, PNG, TGA, TIF, PSD, ICO

Comando CAM1

Activa la vista de una cámara fija.
Cámara representada por una entidad 'CamaraFija' situada en el escenario.

Sintaxis del comando:
ACCI CAM1 número_de_cámara
'número_de_cámara': Número de cámara fija que se desea activar (entre 1 y 64).
Este número es el que asignamos en la casilla 'Num_camara' de la entidad 'CamaraFija' cuando creamos esta cámara.

Ejemplo: Queremos activar la cámara número 2 para que sea la vista que vea el jugador:
ACCI CAM1 2

Consulta la sección del comando 'CAM0' para saber como desactivar la vista de una cámara fija.

Comando CHKP

Establece el último punto 'checkpoint' donde reaparecerá el jugador si pierde una vida en el nivel.

Sintaxis del comando:
ACCI CHKP

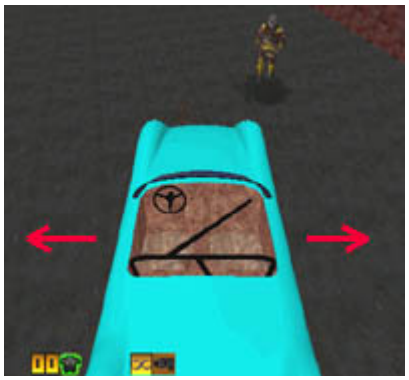
Con este comando podemos establecer ciertos puntos en nuestro escenario que servirán para hacer reaparecer al jugador una vez lo hayan matado, de esta forma no tendrá que volver a comenzar el nivel desde el principio.
Se pueden utilizar distintos comandos 'CHKP' en un mismo nivel, pero sólo el último ejecutado será el que se tomará como punto de reaparición para el jugador.
Al ser 'CHKP' un comando de Entidad 3D lo podemos utilizar en un fichero .ini que se lanzará utilizando cualquier entidad que soporte ficheros de comandos .ini, como son por ejemplo las entidades 'Command' o 'SpecificActor'.
El comando 'CHKP' lo único que realiza es guardar el punto que en ese momento ocupa el jugador para poder hacerlo reaparecer en ese mismo punto cuando pierda una vida, al mismo tiempo que se ejecuta este comando aparecerá la palabra 'CHECKPOINT' superpuesta en pantalla durante unos segundos.
Como ves, de ti dependerá indicar al jugador, dentro del nivel, que está a punto de tocar un 'checkpoint', lo más indicado es representarlo con una entidad 'SpecificActor' y de paso utilizamos el comando 'CHKP' en un fichero .INI asociado a esa entidad.
Al ser 'CHKP' un comando y no una entidad ligada a un punto en concreto del escenario, las posibilidades de establecer puntos de 'Checkpoint' en nuestro juego se multiplican, ya que no sólo se puede establecer un 'checkpoint' en el espacio sino en el tiempo, según circunstancias, según acciones previas del jugador, mediante retardos, según condiciones, etc...
Ejemplo: Queremos establecer un 'checkpoint' cuando el jugador alcance un punto del nivel:

```
IF COND CHK1=0
ACCI CHKP
ACCI PONE CHK1=1
ENDIF
```

Observa que he utilizado una variable, en este caso CHK1, para básicamente controlar dos cosas: para que no se vuelva a activar este 'checkpoint' aunque el jugador vuelva a pasar por ese punto y por otro lado, como el jugador reaparecerá en este punto, si no tenemos este comando 'CHKP' controlado con una variable, cada vez que el jugador volviera a reaparecer volvería a activarse el 'checkpoint'.
El gráfico que se utiliza para mostrar la palabra 'CHECKPOINT', lo podrás encontrar en el fichero 'hud_font2.bmp' dentro de la carpeta 'bmp' de Entidad 3D. Puedes editar este fichero para modificar este gráfico, pero recuerda dejar ese color verde de fondo (color número 250) que hace de color transparente.

Porción del fichero 'hud_font2.bmp' utilizada para el 'CHECKPOINT'

Comandos LAT0 y LAT1



'LAT0' no permitirá al jugador moverse lateralmente (derecha/izquierda).
Se utiliza para no permitir ese movimiento cuando el jugador controle por ejemplo un automóvil, moto, etc.., ya que no quedaría nada realista.
Si en cualquier momento del juego el jugador ya vuelve a controlar un personaje o vehículo que disponga de movimiento lateral, sólo tendremos que volver a permitir el movimiento con el comando **'ACCI LAT1'**.

Sintaxis del comando 'LAT0':
ACCI LAT0
Sintaxis del comando 'LAT1':
ACCI LAT1

Comando CTEX

Permite cambiar una textura del nivel en pleno juego.

Sintaxis del comando:
ACCI CTEX nombre_nueva_textura nombre_textura_nivel
'nombre_nueva_textura': Nombre del fichero de la nueva textura que quieres poner.
No se debe poner la extensión '.BMP'. El fichero de textura debe estar en la carpeta 'bmp\texturas' de Entidad 3D (aquí si debe tener la extensión .BMP) y debe ser una imagen de 255 colores y del mismo tamaño que la textura que se quiere sustituir.
'nombre_textura_nivel': Nombre de la textura que quieres sustituir en el nivel. Es el nombre de la textura que has utilizado en World Editor cuando creaste el nivel y que ahora deseas cambiar. No se debe poner la extensión '.BMP'.

El comando 'CTEX' se puede utilizar para por ejemplo activar una pantalla que estaba apagada, cambiar el aspecto de una habitación, presentar mensajes, cambiar una textura con una luz roja (acceso denegado) por otra con una luz verde (acceso permitido), indicar al jugador por donde seguir, presentar mapas que antes no estaban, etc..
Como ves las posibilidades son infinitas al permitirte poder alterar cuantas texturas desees de tu nivel en función de las acciones del jugador, de temporizadores, de contadores, etc..

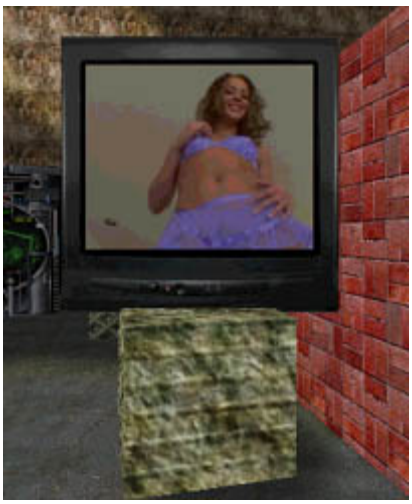
Ejemplo: Queremos activar una pantalla que al iniciarse el nivel no mostraba nada:
ACCI CTEX pantalla1 pantalla0

Textura 'pantalla0' que hemos utilizado en World Editor para una cara del cubo que representa el monitor.

Textura 'pantalla1.bmp', que hemos copiado previamente en la carpeta 'bmp\texturas'.

Resultado inicial y final, en pleno juego.

Comandos ANI1 y ANI0



'ANI1' inicia una animación definida en el nivel con una entidad 'Animacion'.
'ANI0' detiene y hace desaparecer una animación definida en el nivel con una entidad 'Animacion'.

Sintaxis del comando:
ACCI ANI1 número_de_animación
'número_de_animación': Número de animación que se desea activar (entre 1 y 255).
Este número es el que asignamos en la casilla 'Num_animacion' de la entidad 'Animacion' cuando creamos esta animación.

Ejemplo: Queremos activar la animación número 3:
ACCI ANI1 3
Para desactivar cualquier animación, basta con el siguiente comando:
ACCI ANI0

Consulta la sección de la entidad 'Animacion' para ver algunos ejemplos.

Comando LUZ1

Activa una luz colocada con la entidad 'DynamicLight'.

Sintaxis del comando:
ACCI LUZ1 número_de_luz
'número_de_luz': Número de luz que se desea activar (entre 1 y 255).
Este número es el que asignamos en la casilla 'Num_luz' de la entidad 'DynamicLight' cuando creamos esta luz.

Ejemplo: Durante el transcurso del juego queremos activar la luz número 3 para que sea visible.
ACCI LUZ1 3

Consulta la sección del comando 'LUZ0' para saber como desactivar una luz.

Comando LUZ0

Desactiva una luz colocada con la entidad 'DynamicLight'.

Sintaxis del comando:
ACCI LUZ0 número_de_luz
'número_de_luz': Número de luz que se desea desactivar (entre 1 y 255).
Este número es el que asignamos en la casilla 'Num_luz' de la entidad 'DynamicLight' cuando creamos esta luz.

Ejemplo: Durante el transcurso del juego queremos desactivar la luz número 2 para que no sea visible.
ACCI LUZ0 2

Consulta la sección del comando 'LUZ1' para saber como activar una luz.

Comando COR1

Activa una luz corona colocada con la entidad 'Corona'.

Sintaxis del comando:
ACCI COR1 número_de_corona
'número_de_corona': Número de corona que se desea activar (entre 1 y 255).
Este número es el que asignamos en la casilla 'Num_corona' de la entidad 'Corona' cuando creamos esta corona.

Ejemplo: Durante el transcurso del juego queremos activar la luz corona número 4 para que sea visible.
ACCI COR1 4

Consulta la sección del comando 'COR0' para saber como desactivar una luz corona.

Comando COR0

Desactiva una luz corona colocada con la entidad 'Corona'.

Sintaxis del comando:
ACCI COR0 número_de_corona
'número_de_corona': Número de corona que se desea desactivar (entre 1 y 255).
Este número es el que asignamos en la casilla 'Num_corona' de la entidad 'Corona' cuando creamos esta corona.

Ejemplo: Durante el transcurso del juego queremos desactivar la luz corona número 4 para que no sea visible.
ACCI COR0 4

Consulta la sección del comando 'COR1' para saber como activar una luz corona.

Comando TEA1

Activa una textura animada durante el transcurso del juego.
Si la secuencia de la textura estaba detenida, al activarla se pondrá en marcha.

Sintaxis del comando:
ACCI TEA1 secuencia_bmp
'secuencia_bmp': Nombre de la secuencia de ficheros .bmp que pusimos en el fichero .PRC para la textura que queremos activar.

Ejemplo: Durante el transcurso del juego queremos activar la textura 'cascada'.
ACCI TEA1 cascada

Consulta la sección de texturas animadas para saber como se definen las texturas animadas en un nivel mediante ficheros .PRC.
Consulta la sección del comando TEA0 para saber como desactivar una textura animada durante el transcurso del juego.

Comando TEA0

Desactiva una textura animada durante el transcurso del juego.
Si la secuencia de la textura estaba en marcha, al desactivarla se detendrá.

Sintaxis del comando:
ACCI TEA0 secuencia_bmp
'secuencia_bmp': Nombre de la secuencia de ficheros .bmp que pusimos en el fichero .PRC para la textura que queremos desactivar.

Ejemplo: Durante el transcurso del juego queremos desactivar la textura 'plasma'.
ACCI TEA0 plasma

Consulta la sección de texturas animadas para saber como se definen las texturas animadas en un nivel mediante ficheros .PRC.
Consulta la sección del comando TEA1 para saber como activar una textura animada durante el transcurso del juego.

Comando QSAL

Permite quitar una determinada cantidad de salud al jugador.

Sintaxis: **ACCI QSAL cantidad_de_salud**
'cantidad_de_salud': Valor que determina la cantidad de salud que se le restará al jugador.
Se permiten valores de 1 a 100.
Ejemplo: Bajo cierta circunstancia decidimos restar 20 puntos de salud al jugador:
ACCI QSAL 20

Comando DSAL

Permite incrementar una determinada cantidad de salud al jugador.

Sintaxis: **ACCI DSAL cantidad_de_salud**
'cantidad_de_salud': Valor que determina la cantidad de salud que se le aumentará al jugador.
Se permiten valores de 1 a 100.
Ejemplo: Bajo cierta circunstancia decidimos premiar con 15 puntos de salud al jugador:
ACCI DSAL 15

Comando MOV0

Comando para bloquear el movimiento del jugador durante un tiempo o hasta que se utilice el comando MOV1.

Sintaxis: **ACCI MOV0 tiempo**
'tiempo': Tiempo en segundos que el jugador no podrá desplazarse.
Poner 0 para un tiempo indefinido.

Ejemplo: Queremos bloquear al jugador durante 15 segundos:
IF COND BL=0
ACCI MOV0 15
ACCI PONE BL=1
ENDIF
Observa que he utilizado una variable, en este caso BL, para que no se vuelva a activar este bloqueo aunque el jugador vuelva a pasar por ese punto, ya que como no se mueve se volvería a activar el bloqueo y no saldríamos nunca de ese punto.
Si queremos desbloquear al jugador antes de que finalice el tiempo de bloqueo establecido, podemos utilizar el comando MOV1:
ACCI MOV1

Comando TELETRANSPORTE

Transporta al jugador, de forma instantanea, a otra posición del nivel.

Sintaxis: **ACCI TELETRANSPORTE número_de_posición**
'número_de_posicion': Número de posición a la que se quiere enviar al jugador (entre 1 y 255).
Este número es el que asignamos en la casilla 'Num_posicion' de alguna entidad 'Posicion' que hayamos colocado en nuestro nivel.

Ejemplo:
Queremos teletransportar al jugador cuando pase por un punto en contreto:
En ese punto situamos una entidad '**Command**' y le asignamos un fichero de comandos que llamaremos por ejemplo: 'teleport5.ini'.
Situamos una entidad '**Posicion**' con número de posición **5** en la posición donde queremos enviar al jugador.
Si no queremos condicionar el teletransporte a nada y que siempre se teletransporte el jugador, el fichero 'teleport5.ini' debería contener simplemente lo siguiente:
ACCI TELETRANSPORTE 5

---

*» El fichero de sonido que usa el comando 'TELETRANSPORTE' se llama '**teletransporte.wav**' y está en la carpeta '**WAV**' de Entidad 3D.*

Comando PAN1

Añade al panel de objetivos uno de los objetivos que hemos definido en nuestro nivel con la entidad 'Objetivo'.

Sintaxis del comando:
ACCI PAN1 número_de_objetivo
'número_de_objetivo': Número de objetivo que le hemos asignado en el parámetro 'Num_objetivo' de la entidad 'Objetivo' para este objetivo en concreto.

Ejemplo: Durante el transcurso del juego queremos añadir al panel de objetivos de la misión el objetivo número 3:
ACCI PAN1 3

Para borrar objetivos del panel o darlos por finalizados, utiliza los comandos 'PAN0' y 'OBJ1'.

Comando PAN0

Oculta objetivos del panel de objetivos de la misión. Este objetivo debe haber sido definido en nuestro nivel con la entidad 'Objetivo'.

Sintaxis del comando:
ACCI PAN0 número_de_objetivo
'número_de_objetivo': Número de objetivo que le hemos asignado en el parámetro 'Num_objetivo' de la entidad 'Objetivo' para este objetivo en concreto.

Ejemplo: Durante el transcurso del juego queremos ocultar del panel de objetivos de la misión el objetivo número 2, ya sea porque está cumplido, porque ha finalizado el nivel, etc.. :
ACCI PAN0 2

Para añadir objetivos al panel o darlos por finalizados, utiliza los comandos 'PAN1' y 'OBJ1'.

Comando OBJ1

Establece como cumplido uno de los objetivos de la misión que hemos definido en nuestro nivel con la entidad 'Objetivo'.
En el panel se reflejará con una marca verde en la casilla de ese objetivo.

Sintaxis del comando:
ACCI OBJ1 número_de_objetivo
'número_de_objetivo': Número de objetivo que le hemos asignado en el parámetro 'Num_objetivo' de la entidad 'Objetivo' para este objetivo en concreto.

Ejemplo: Durante el transcurso del juego queremos dar por cumplido el objetivo número 5 de la misión::
ACCI OBJ1 5

Para ocultar o añadir objetivos al panel, utiliza los comandos 'PAN0' y 'PAN1'

Comando INPU

Se utiliza para permitir al jugador, mediante la consola, entrar por teclado algún tipo de texto durante el juego.
Resulta útil para pedir claves de acceso o respuestas concretas al jugador.

Sintaxis: **ACCI INPU texto_pregunta=fichero_de_comandos**
'texto_pregunta': Es un texto con la pregunta que deseamos formular al jugador.
Esta pregunta aparecerá en la consola cuando se active el comando 'INPU'.
'fichero_de_comandos': Nombre del fichero de comandos que se procesará una vez el jugador haya introducido el texto.

El funcionamiento de este comando una vez ejecutado es el siguiente:
1º Hace aparecer de forma automática la consola.
2º En la consola aparece el texto de la pregunta que hemos indicado en el comando 'INPU'.
3º El programa espera que el jugador entre algo y pulse 'Enter'. No se le permite salir de la consola ni pulsando 'ESC' ni pulsando 'TAB' ni de ninguna otra forma que no sea entrando algo y pulsando 'Enter'.
4º Una vez el jugador ha pulsado 'Enter', todo lo que haya entrado se guarda en la variable INPUT.
5º Se procesa el fichero de comandos que hemos indicado en el comando 'INPU'.
Como ves, de nosotros depende comprobar en un fichero de comandos si el jugador ha introducido lo que nos interesa y actuar en consecuencia con los comandos que nos parezcan oportunos.

Ejemplo:
Pedir una 'password' al jugador y presentar en pantalla si es correcta o no. No dejaremos moverse al jugador hasta que conteste algo. La pregunta se la haremos siempre que pase por un lugar en concreto mientras no conteste correctamente.
Creamos un fichero de comandos llamado 'pregunta.ini' que asignaremos en una entidad 'Command' que situaremos en el lugar que queremos formular la pregunta.
Este sería el contenido del fichero 'pregunta.ini':

```
IF COND PREGUNTANDO=0
ACCI PONE PREGUNTANDO=1
ACCI MOV0
ACCI INPU Introduce password:=comprueba.ini
ENDIF
```

También crearemos otro fichero de comandos llamado 'comprueba.ini' que es el que se procesa una vez el jugador ha introducido su respuesta y es el mismo fichero de comandos que hemos indicado en el comando 'INPU'.
Este sería el contenido del fichero 'comprueba.ini' que responde '***CORRECTO***' al contestar la palabra 'ZATRAS':

```
ACCI TEXT Comprobando password ...
IF COND INPUT=ZATRAS
ACCI TEXT ***CORRECTO***
ELSE
ACCI TEXT INCORRECTO!!!
ACCI PONE PREGUNTANDO=0
ENDIF
ACCI MOV1
```

Comando NIE0

Este comando desactiva la niebla del escenario si se activó previamente con el comando 'NIE1' o bien si estaba activada al dar comienzo el nivel.

Sintaxis del comando:
ACCI NIE0

Ejemplo: Queremos **desactivar** el efecto de niebla de nuestro nivel, en un punto donde el jugador se dispone a entrar en un edificio, ya que hasta ese momento el jugador se encontraba en un exterior donde estaba activado el efecto de niebla.
Dentro de un pasillo, en el extremo del pasillo que da al interior del edificio, colocamos una entidad 'Command' con un fichero de comandos .INI asociado que contendría lo siguiente:
ACCI NIE0

» Consulta la sección del comando 'NIE1' para saber como volver a activar el efecto de niebla.
» Para poder usar los comandos 'NIE0' y 'NIE1' tienes que haber situado en tu nivel una entidad 'Fx-Niebla'.

Comando NIE1

Este comando activa la niebla del escenario si se desactivó previamente con el comando 'NIE0' o bien si estaba desactivada al dar comienzo el nivel.

Sintaxis del comando:
ACCI NIE1

Ejemplo: Queremos **activar** el efecto de niebla de nuestro nivel, en un punto donde el jugador se dispone a salir de en un edificio al exterior, ya que hasta ese momento el jugador se encontraba en un interior donde estaba desactivado el efecto de niebla.
Dentro de un pasillo, en el extremo del pasillo que da al exterior, colocamos una entidad 'Command' con un fichero de comandos .INI asociado que contendría lo siguiente:
ACCI NIE1

» Consulta la sección del comando 'NIE0' para saber como volver a activar el efecto de niebla.
» Para poder usar los comandos 'NIE0' y 'NIE1' tienes que haber situado en tu nivel una entidad 'Fx-Niebla'.

Comando ARMAENERGIA

Activa una arma de energía alrededor del jugador, dañando a todos los enemigos que alcance.

Sintaxis del comando:
ACCI ARMAENERGIA número_de_arma_de_energía
'número_de_arma_de_energía': Número de arma de energía que se desea activar (entre 1 y 20).
Este número es el que asignamos en la casilla 'Num_ArmaEnergia' de la entidad 'Fx_ArmaEnergia' cuando creamos esta arma.

Ejemplo 1:
Queremos hacer que cuando el jugador use el objeto '18_arma3.act' (objeto número 18) de su inventario de objetos, se active la arma número 3 alrededor del jugador:
IF COND ACCION_SELECCIONADA=Usar COND OBJETO_SELECCIONADO=18
ACCI ARMAENERGIA 3
ACCI QINVUSO 18_arma3.act
ENDIF
Como ves, utilizando el sistema de **menú de acción**+**objeto**, te permite controlar muy fácilmente la cantidad de usos que el jugador puede realizar de una arma de energía en concreto.
Para que el ejemplo anterior funcione en cualquier momento y lugar del nivel, las líneas de comandos anteriores tendrías que incluirlas en el fichero de comandos .INI general que se indica en el parámetro '**FicheroAcciones**' de la entidad 'Configuracion'.

Ejemplo 2:
Queremos hacer que cuando el jugador pulse la tecla '**U**' se active la arma número 4 alrededor del jugador:
Añadiríamos a nuestro nivel una entidad TeclaComandos con los valores siguientes:
FicheroComandos = **activa_arma4.ini**
Tecla = **U**
El fichero de comandos '**activa_arma4.ini**' simplemente contendría:
ACCI ARMAENERGIA 4

Ejemplo 3:
Queremos hacer lo mismo que el ejemplo anterior, pero sólo permitiendo que el jugador use esta arma 10 veces.
El fichero de comandos '**activa_arma4.ini**' contendría:
IF COND USOS_ARMA4>0
ACCI ARMAENERGIA 4
ACCI OPE- USOS_ARMA4 1
ENDIF
Para entregar al jugador la arma con las 10 cargas tendríamos que hacer lo siguiente en algún momento del juego:
ACCI PONE USOS_ARMA4=10

---

*» Consulta la entidad 'Fx_ArmaEnergia' para saber como definir una arma de energía antes de utilizar el comando '**ARMAENERGIA**'.*

Comando SALTO

Permite variar, durante el juego, la altura a la que puede saltar el jugador.

Sintaxis: **ACCI SALTO altura tiempo**
'altura': Valor que determina la máxima altura a la que podrá saltar el jugador cuando use la tecla o botón de salto.
Se permiten valores de 0 a 5000.
'tiempo': Tiempo, en segundos, que el jugador podrá saltar a la nueva altura, después de ese tiempo sólo podrá saltar a la altura original.
Deja a 0 este valor para un tiempo indefinido.

Ejemplo 1: Permitir saltar al jugador a una altura de 1500, pero sólo durante 15 segundos:
ACCI SALTO 1500 15
Ejemplo 2: Permitir saltar al jugador a una altura de 1000 sin límite de tiempo:
ACCI SALTO 1000 0

---

*» Consulta la sección de la entidad 'DeathMatchStart' parámetro '**Salto_jugador**' para establecer la altura de salto inicial.*

Comando VELOCIDAD

Permite variar, durante el juego, la velocidad a la que se desplaza el jugador.

Sintaxis: **ACCI VELOCIDAD velocidad tiempo**
'velocidad': Valor que determina la velocidad a la que se desplazará el jugador por el nivel.
Se permiten valores de 0 a 10000.
'tiempo': Tiempo, en segundos, que el jugador podrá desplazarse a la nueva velocidad, después de ese tiempo sólo podrá desplazarse a la velocidad original.
Deja a 0 este valor para un tiempo indefinido.

Ejemplo 1: Permitir al jugador desplazarse a una velocidad de 5500, pero sólo durante 20 segundos:
ACCI VELOCIDAD 5500 20
Ejemplo 2: Permitir al jugador desplazarse a una velocidad de 4900 sin límite de tiempo:
ACCI VELOCIDAD 4900 0

---

*» Consulta la sección de la entidad 'DeathMatchStart' parámetro '**Velocidad**' para establecer la velocidad inicial.*

Comando SALIRJUEGO

Permite salir de tu juego limpiamente y sin presentar ningún tipo de mensaje.

Sintaxis: **ACCI SALIRJUEGO tiempo nivel aplicación**
'tiempo': Retardo de tiempo, en segundos, antes de proceder a salir del juego.
Deja a 0 este valor si no quieres que haya retardo.
'nivel': Nombre del próximo nivel del juego para salvarlo en la carpeta de partidas. De esta forma el jugador podrá usar la opción 'cargar partida' para empezar el juego en el nivel que hayamos indicado.
Para este parámetro, indicar la palabra **nada** en el caso de que no quieras salvar partida.
'aplicación': Nombre de alguna aplicación de Windows que se abrirá justo después de salir del juego.
Para este parámetro, indicar la palabra **nada** en el caso de que no quieras abrir ninguna aplicación.

Este comando permite controlar el lanzamiento de niveles de Entidad 3D desde una aplicación externa, como podría ser un programa .EXE realizado con aplicaciones como Macromedia Flash. Desde esa aplicación se podrían lanzar los niveles que se seleccionen en un menú y desde el juego, una vez finalizado en nivel, volver otra vez al menú usando el comando '**SALIRJUEGO**'.

Ejemplo 1: Salir del juego y ejecutar la aplicación **Notepad** de Windows:
ACCI SALIRJUEGO 0 nada notepad.exe
Ejemplo 2: Esperar 5 segundos, salir del juego y ejecutar la aplicación **menu.exe**. Se salva partida para volver a empezar el juego en el nivel llamado 'torre' al usar la opción 'cargar partida':
ACCI SALIRJUEGO 5 torre menu.exe
Ejemplo 3: Simplemente salir del juego:
ACCI SALIRJUEGO 0 nada nada
Ejemplo 4:
Hasta se podría lanzar un nuevo nivel de esta forma:
ACCI SALIRJUEGO 0 nivel2 Entidad3d.exe -name Jugador -map nivel2 -PickMode

Comando EXEC

Permite lanzar un programa externo durante el transcurso del juego.

Sintaxis: **ACCI EXEC aplicación,título_ventana,modo**
'aplicación': Nombre de la aplicación de Windows que deseamos lanzar.

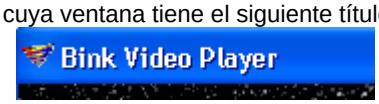
'título_ventana': Nombre del título de la ventana que crea la aplicación que vamos a lanzar. Este nombre es el que aparece en la parte superior de la ventana de la aplicación cuando está funcionando.
'modo': Se usa para establecer de que modo se abrirá la aplicación. Tenemos varios modos:
- Para modo normal pondremos el número **1**. Este modo abre la aplicación sin forzarla a que se maximice y por lo tanto se abrirá de la manera que la aplicación tenga por defecto.
- Para modo maximizado pondremos el número **3**. En este modo se intentará abrir la aplicación de modo maximizado (no siempre lo soporta la aplicación) para que ocupe el mayor tamaño posible en pantalla.

Este comando permite ejecutar programas externos a Entidad 3D en cualquier momento del desarrollo de nuestro juego. Esto nos puede servir para presentar videos en momentos determinados del juego (en formato autoejecutable .EXE) o para lanzar cualquier programa que se nos ocurra.
Cuando usemos este comando, Entidad 3D permanecerá minimizado en la barra de tareas de Windows mientras esté funcionando la aplicación que hayamos lanzado. Una vez esta finalice (porque la ha cerrado el jugador, porque ella misma se cierra automáticamente o por la razón que sea), Entidad 3D volverá a ponerse automáticamente a pantalla completa y continuará con lo que estaba haciendo antes de lanzar la aplicación.

Ejemplo 1: Lanzar el programa '**videoflash.exe**' cuya ventana tiene el siguiente título:

... estas serían las líneas de comandos:
IF COND VIDEO=0
ACCI PONE VIDEO=1
ACCI EXEC videoflash.exe,Macromedia Flash Player 6,3
ENDIF
En este ejemplo, el programa '**videoflash.exe**' debe estar en la carpeta principal de Entidad 3D, si lo ponemos en otra carpeta deberemos indicar la ruta completa en el comando EXEC.
Usamos la variable **VIDEO** (que puede ser cualquier nombre de variable que nos inventemos) para controlar que no se lance la aplicación múltiples veces seguidas.

Ejemplo 2: Lanzar el programa '**video1.exe**' cuya ventana tiene el siguiente título:



... estas serían las líneas de comandos:
IF COND VIDEOX=0
ACCI PONE VIDEOX=1
ACCI EXEC videos/video1.exe,Bink Video Player,1
ENDIF
En este ejemplo, el programa '**video1.exe**' debe estar en una carpeta llamada **videos** que debe partir de la carpeta principal de Entidad 3D.

» *Algunos programas que puedes usar para crear ficheros autoejecutables .EXE conteniendo videos o presentaciones, pueden ser diversas herramientas 'Macromedia Flash' de Adobe o 'Bink Video' de 'RAD game tools'.*

Comando BARRAACTIVAR

Activa un barra que haya sido desactivada con el comando BARRADESACTIVAR o que esté desactivada desde el comienzo del nivel.

Sintaxis del comando:
ACCI BARRAACTIVAR número_de_barra
'número_de_barra': Número de barra que se desea activar (entre 1 y 10).
Este número es el que asignamos en la casilla 'Num_barra' de la entidad 'MarcadorBarra' cuando creamos esta barra.

Una barra activada se llenará o se vaciará a medida que pase el tiempo, siempre y cuando esté a 'True' el parámetro 'Tiempo' de la entidad 'MarcadorBarra' que define esta barra.
Ejemplo
Activar la barra número 1:
ACCI BARRAACTIVAR 1

Comando BARRADESACTIVAR

Desactiva un barra que haya sido activada con el comando BARRAACTIVAR o que esté activada desde el comienzo del nivel.

Sintaxis del comando:
ACCI BARRADESACTIVAR número_de_barra
'número_de_barra': Número de barra que se desea desactivar (entre 1 y 10).

Este número es el que asignamos en la casilla 'Num_barra' de la entidad 'MarcadorBarra' cuando creamos esta barra.

Una barra desactivada no se llenará ni se vaciará a medida que pase el tiempo, aunque esté a 'True' el parámetro 'Tiempo' de la entidad 'MarcadorBarra' que define esta barra.

Ejemplo
Desactivar la barra número 1:
ACCI BARRADESACTIVAR 1

Comando BARRAVISIBLE

Hace visible un barra que esté invisible por el uso del comando BARRAINVISIBLE o que sea invisible desde el comienzo del nivel.

Sintaxis del comando:
ACCI BARRAVISIBLE número_de_barra
'número_de_barra': Número de barra que se desea hacer visible (entre 1 y 10).
Este número es el que asignamos en la casilla 'Num_barra' de la entidad 'MarcadorBarra' cuando creamos esta barra.

Ejemplo
Hacer visible la barra número 2:
ACCI BARRAVISIBLE 2

Comando BARRAINVISIBLE

Hace invisible un barra que esté visible por el uso del comando BARRAVISIBLE o que sea visible desde el comienzo del nivel.

Sintaxis del comando:
ACCI BARRAINVISIBLE número_de_barra
'número_de_barra': Número de barra que se desea hacer invisible (entre 1 y 10).
Este número es el que asignamos en la casilla 'Num_barra' de la entidad 'MarcadorBarra' cuando creamos esta barra.

Ejemplo
Hacer invisible la barra número 2:
ACCI BARRAINVISIBLE 2

Comando BARRAVALOR

Permite, durante el transcurso del juego, modificar el valor de una barra en concreto para variar su nivel de llenado.

Sintaxis del comando:
ACCI BARRAVALOR número_de_barra valor
'número_de_barra': Número de barra que deseamos modificar su valor (entre 1 y 10).
Este número es el que asignamos en la casilla 'Num_barra' de la entidad 'MarcadorBarra' cuando creamos esta barra.
'valor': Como mínimo puede valer **1** y como máximo depende del tamaño de la barra.

Ejemplo
Asignar el valor 40 a la barra número 3:
ACCI BARRAVALOR 3 40

Comando SETCLAVEJUEGO

Establece la clave del juego para funciones de protección del juego.
Junto con el comando CHKCLAVEJUEGO, sirve para evitar que el jugador empiece a jugar por otro nivel que no sea por el primero.

Sintaxis del comando:
ACCI SETCLAVEJUEGO clave
'clave': Un texto que debes introducir (inventado) para establecer la clave de tu juego.
Este texto puede ser en realidad una cadena de letras y/o números (64 caracteres como máximo).

Ejemplo
Establecer la clave '**Nemesis**' como clave de juego:
ACCI SETCLAVEJUEGO Nemesis
Sitúa esa línea al principio del fichero de comandos que uses como fichero .INI inicial de tu juego. Básicamente en el fichero que indicas en el parámetro '**FicheroComandos**' de la entidad 'Configuración' que habrás puesto en el primer nivel de tu juego. Si no usas ese fichero de comandos es conveniente que lo crees y lo indiques en el citado parámetro.
Pero con sólo situar la línea '**ACCI SETCLAVEJUEGO Nemesis**' en este fichero de comandos no es suficiente, para más seguridad sitúa la misma línea en algunos otros ficheros .INI (en diversos sitios, no siempre al principio) que use tu juego en el

primer nivel. Esto es para evitar que el jugador no pueda prescindir de tus ficheros .INI ya que si sólo indicas el '**ACCI SETCLAVEJUEGO Nemesis**' en el fichero inicial, este fichero podría ser reemplazado por otro con una clave distinta (o sin clave) para intentar saltarse la protección.
Evidentemente no uses la clave '**Nemesis**' para tu juego, sino cualquiera otra que te inventes.
Esta clave de juego también la utiliza el programa de forma automática y de manera interna para otras labores de protección del juego.

¿ Cómo funciona la protección ?:
Si el jugador intenta comenzar a jugar, por ejemplo por el segundo nivel, y en ese nivel se usan ficheros .INI que contienen la línea de comandos ACCI CHKCLAVEJUEGO Nemesis, el programa comprobará si la clave del juego es 'Nemesis', como no se ha establecido la clave del juego (ya que eso sólo lo hace en el primer nivel con '**ACCI SETCLAVEJUEGO Nemesis**'), pues el juego dejará de funcionar y presentará un mensaje de error.
¿ Y si el jugador mira mis ficheros .INI para saber mi clave de juego ? ... Se supone que en la distribución definitiva de tu juego estás usando ficheros '**ini.enc**' encriptados que contienen información ilegible para el jugador, pero entendible para el programa.

---

*» Recuerda que este comando sólo lo debes usar en ficheros .INI del **primer nivel** de tu juego.*
*» Utiliza **la misma clave de juego** en todos los comandos SETCLAVEJUEGO y CHKCLAVEJUEGO que utilices para un juego en concreto.*
*» Si usas el comando SALIRJUEGO para controlar el lanzamiento de los niveles de tu juego, este sistema de situar **SETCLAVEJUEGO** en ficheros .INI del primer nivel y **CHKCLAVEJUEGO** en ficheros .INI de los niveles posteriores, no te funcionará, ya que al salir del primer nivel devuelves el control a Windows y la clave de juego establecida se pierde. Si de todas formas no te importa que el jugador pueda comenzar por el nivel que quiera pero quieres proteger los ficheros de nivel .BSP para que no sean manipulados, sólo tendrás que usar comandos **SETCLAVEJUEGO** en diferentes .INI de todos los niveles de tu juego, no te harán falta los CHKCLAVEJUEGO.*

Comando CHKCLAVEJUEGO

Comprueba que la clave de juego establecida es la que se indica a continuación de este comando.
Junto con el comando SETCLAVEJUEGO, sirve para evitar que el jugador empiece a jugar por otro nivel que no sea por el primero.

Sintaxis del comando:
ACCI CHKCLAVEJUEGO clave
'clave': Es el mismo texto que usaste para el comando SETCLAVEJUEGO para establecer la clave de tu juego en el primer nivel.
Este texto puede ser en realidad una cadena de letras y/o números (64 caracteres como máximo).

Ejemplo
Comprobar si la clave de juego establecida es la clave '**Nemesis**':
ACCI CHKCLAVEJUEGO Nemesis
Sitúa esa línea en diferentes ficheros de comandos .INI que use tu juego en todos los niveles posteriores al primero (en el segundo, en el tercero, etc...). Puedes poner esta línea en cualquier sitio, al principio de los .INI, en medio, al final, etc... El caso es asegurarte de que al menos un comando CHKCLAVEJUEGO se procesará para comprobar que la clave de juego es la correcta.

---

*» Recuerda que este comando sólo lo debes usar en ficheros .INI que se usen en el **segundo nivel y posteriores**.*
*» Utiliza **la misma clave de juego** en todos los comandos SETCLAVEJUEGO y CHKCLAVEJUEGO que utilices para un juego en concreto.*

Comando ENERGIAVOLAR

Permite, durante el transcurso del juego, incrementar o reducir la energía de vuelo del jugador.

Sintaxis del comando:
ACCI ENERGIAVOLAR valor
'valor': Como mínimo puede valer **0** y como máximo **100**.
La energía de vuelo del jugador se incrementará con el valor que se haya indicado.
Usar valores negativos para reducir la energía de vuelo.

Ejemplo 1
Incrementar 15 puntos la energía de vuelo del jugador:
ACCI ENERGIAVOLAR 15

Ejemplo 2
Reducir 10 puntos la energía de vuelo del jugador:
ACCI ENERGIAVOLAR -10

» Consulta la entidad 'VolarJugador' para saber como configurar el modo volar del jugador.

Comando JUGADORVOLAR1

Activa el modo volar del jugador.

Sintaxis del comando:
ACCI JUGADORVOLAR1
Ejemplo: Queremos hacer que el jugador pueda volar porque lo hemos convertido en una águila:
ACCI JUGADORVOLAR1

» Usa el comando 'JUGADORVOLAR0' para desactivar el modo volar del jugador.
» Consulta la entidad 'VolarJugador' para saber como configurar el modo volar del jugador.

Comando JUGADORVOLAR0

Desactiva el modo volar del jugador.

Sintaxis del comando:
ACCI JUGADORVOLAR0
Ejemplo: Queremos hacer que el jugador no pueda volar porque lo hemos convertido en un ser humado (antes era una águila):
ACCI JUGADORVOLAR0

» Usa el comando 'JUGADORVOLAR1' para activar el modo volar del jugador.
» Consulta la entidad 'VolarJugador' para saber como configurar el modo volar del jugador.

Comando SALVAVARIABLE

Permite salvar en un fichero el valor de una variable. Posteriormente, esta variable se podrá recuperar usando el comando CARGAVARIABLE.

Sintaxis: **ACCI SALVAVARIABLE sección variable fichero**
'sección': Nombre de la sección, dentro del fichero, donde queremos salvar la variable junto con su valor.
'variable': Nombre de la variable que queremos salvar.
'fichero': Nombre del fichero donde queremos salvar la variable.
Este fichero debe estar en la carpeta '**Levels**' de Entidad 3D.
Si el fichero no existe, el propio comando se encargará de crearlo.

```
[fase1]
discos=8


[casa]
discos=5
mensaje1=Gracias por volver


[bar]
botellas=23
LIBROS=Diccionario Manual
mensaje9=Donde estabas ?
```

Fichero **datosnivel1.ini**
Ejemplo de un fichero con secciones (entre corchetes) y las variables con sus valores que pertenecen a cada sección.

Ejemplo 1
Salvar el valor de la variable '**animales**' de la sección '**fase1**' del fichero '**datosnivel1.ini**':
ACCI PONE animales=perro gato caballo
ACCI SALVAVARIABLE fase1 animales datosnivel1.ini

Ejemplo 2
Actualizar el valor de la variable '**botellas**' de la sección '**bar**' del fichero '**datosnivel1.ini**':
ACCI SALVAVARIABLE bar botellas datosnivel1.ini

» Consulta el comando CARGAVARIABLE para saber como recuperar variables de un fichero.

Comando CARGAVARIABLE

Permite recuperar el valor de una variable desde un fichero. Previamente, esta variable tiene que haber sido salvada usando el comando SALVAVARIABLE.

Sintaxis: **ACCI CARGAVARIABLE sección variable fichero**
'sección': Nombre de la sección, dentro del fichero, donde se encuentra la variable cuyo valor queremos recuperar.
'variable': Nombre de la variable cuyo valor queremos recuperar.
'fichero': Nombre del fichero donde se encuentra la variable que nos interesa.
Este fichero debe estar en la carpeta '**Levels**' de Entidad 3D.

```
[fase1]
discos=8


[casa]
discos=5
mensaje1=Gracias por volver


[bar]
botellas=23
LIBROS=Diccionario Manual
mensaje9=Donde estabas ?
```

Fichero **datosnivel1.ini**
Ejemplo de un fichero con secciones (entre corchetes) y las variables con sus valores que pertenecen a cada sección.

Ejemplo 1
Cargar el valor de la variable '**discos**' de la sección '**casa**' del fichero '**datosnivel1.ini**' y presentar el valor en pantalla:

```
ACCI CARGAVARIABLE casa discos datosnivel1.ini
ACCI TEXT %DISCOS%
```

Ejemplo 2
Cargar el valor de la variable '**libros**' de la sección '**bar**' del fichero '**datosnivel1.ini**' y presentar el valor en pantalla:

```
ACCI CARGAVARIABLE bar libros datosnivel1.ini
ACCI TEXT %LIBROS%
```

---

*» Si el comando 'CARGAVARIABLE' no encuentra la variable, cargará la variable con el valor **0**.*
» Consulta el comando SALVAVARIABLE para saber como guardar variables en un fichero.

Carpetas, ficheros de imágenes 2D y sonidos:

Imágenes de introducción

Al dar comienzo nuestro juego podremos realizar una introducción mediante una secuencia de imágenes fijas.

Para realizar esto deberemos copiar las imágenes que nos interesa presentar en la carpeta 'BMP' de 'Entidad 3D'. Los ficheros tendrán que tener estos nombres:
*PrimerNivel*_intro1.bmp
*PrimerNivel*_intro2.bmp
*PrimerNivel*_intro3.bmp
*PrimerNivel*_intro4.bmp
*PrimerNivel*_intro5.bmp
*PrimerNivel*_intro6.bmp
*PrimerNivel*_intro7.bmp
*PrimerNivel*_intro8.bmp
*PrimerNivel*_intro9.bmp
*PrimerNivel*_intro10.bmp
*PrimerNivel*_intro11.bmp
Etc.
*PrimerNivel*_intro99.bmp
Estas imágenes pueden ser archivos gráficos en los siguientes formatos:
BMP, JPG, GIF, LBM, IFF, PBM, PCX, PGM, PNG, TGA, TIF, PSD, ICO
Puedes usar todos los ficheros del mismo formato gráfico o una combinación de distintos formatos gráficos para crear tu presentación.
Por ejemplo, para un juego cuyo primer nivel se llame '**templo**' estos podrían ser los nombres de los ficheros gráficos para una presentación de 5 imágenes:
templo_intro1.bmp
templo_intro2.jpg
templo_intro3.gif
templo_intro4.pcx
templo_intro5.jpg
Para que la presentación se vea correctamente en la mayoría de las resoluciones de pantalla, las dimensiones de las imágenes han de ser de una proporción 4:3, es decir, de tamaños igual a resoluciones estándar de pantalla:
640x480
800x600
1024x768
1280x960
1400x1050
1600x1200
1856x1392
1920x1440
2048x1536
Si mantienes esas dimensiones no importa si en la presentación mezclas distintas dimensiones, en la pantalla siempre aparecerán las imágenes a pantalla completa, sea la resolución de pantalla superior o inferior a la de la imagen.
De todas formas ten en cuenta que cuanta más diferencia haya entre la dimensión de una imagen en concreto y el tamaño de la resolución de pantalla más tiempo tardará Entidad 3D en reescalar esa imagen. Son pocos segundos en un potente PC, pero has de considerar que tu juego puede que lo utilicen en máquinas menos potentes. Yo te recomendaría que usaras imágenes de una resolución de 800x600, tienen la suficiente calidad y Entidad 3D apenas tarda en reescalarlas a 1024x768 o 640x480.
Puede haber hasta un máximo de 99 imágenes de presentación.
Justo cuando el programa de comienzo, se comenzará a ver en pantalla la primera imagen de presentación. Se podrán ir pasando una tras otra mediante un 'click' del botón izquierdo del ratón, pulsando la barra de espacio o pulsando 'Enter'. Para salir de la presentación basta con pulsar la tecla 'Esc' o un 'click' del botón derecho del ratón.
Si sólo nos interesa presentar por ejemplo 3 imágenes, sólo copiaremos 3 imágenes con estos nombres en la carpeta 'BMP'.
Si no queremos presentar ninguna imagen, no copiaremos ninguna imagen con esos nombres en la carpeta 'BMP'.

» Consulta la sección 'Música de introducción' para ver como acompañar esta presentación con un fondo musical.
» Consulta la sección 'Entidad3d.INI' para saber como poner un retardo de tiempo entre las imágenes de presentación.

Imagen inicio del nivel

Justo antes de cargar un nivel, podemos presentar una imagen que permanecerá en pantalla hasta que el jugador pulse el botón izquierdo del ratón, la tecla de espacio o la tecla 'Enter'.
Esta imagen se puede utilizar para introducir al jugador en la temática de ese nivel, para indicar en que zona del juego se encuentra o simplemente para mostrar una portada artística que tenga que ver con el nivel.

Para realizar esto deberemos copiar en la carpeta 'BMP' las imágenes que nos interesa presentar en cada nivel (sólo una por nivel). Los ficheros tendrán que tener este nombre:
*Nivel*_portada.bmp
Nivel deberemos sustituirlo por el nombre del nivel que queremos que aparezca la imagen.
Estas imágenes pueden ser archivos gráficos en los siguientes formatos:
BMP, JPG, GIF, LBM, IFF, PBM, PCX, PGM, PNG, TGA, TIF, PSD, ICO
Como puedes ver, podrás definir una imagen distinta para cada nivel de tu juego.
Por ejemplo, para un nivel llamado '**casa**' podríamos poner este fichero de imagen como portada:
casa_portada.jpg

Consulta la sección 'Imágenes de introducción', los consejos que se dan sobre las dimensiones de las imágenes también son aplicables a estas imágenes de portada de nivel.

Imagen mientras se carga el nivel

Mientras se carga un nivel, podemos presentar una imagen que permanecerá en pantalla hasta que el nivel comience.
Normalmente la carga de un nivel requiere pocos segundos pero de esta forma podremos personalizar un poco más nuestro juego con una imagen distinta para cada nivel, además de evitar que al jugador le aparezca el aburrido mensaje 'Cargando nivel ...'

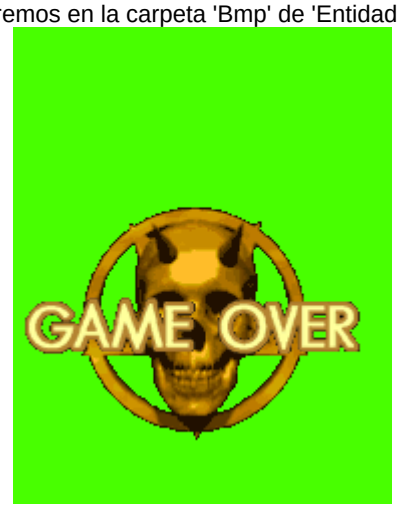
Para realizar esto deberemos copiar en la carpeta 'BMP' las imágenes que nos interesa presentar durante la carga de cada nivel (sólo una por nivel). Los ficheros tendrán que tener este nombre:
*Nivel*_cargando.bmp
Nivel deberemos sustituirlo por el nombre del nivel que queremos que aparezca la imagen.
Estas imágenes pueden ser archivos gráficos en los siguientes formatos:
BMP, JPG, GIF, LBM, IFF, PBM, PCX, PGM, PNG, TGA, TIF, PSD, ICO
Como puedes ver, podrás definir una imagen distinta para cada nivel de tu juego.
Por ejemplo, para un nivel llamado '**iglesia**' podríamos poner este fichero de imagen para que aparezca mientras se carga este nivel:
iglesia_cargando.gif

Consulta la sección 'Imágenes de introducción', los consejos que se dan sobre las dimensiones de las imágenes también son aplicables a estas imágenes de carga de nivel.

Imagen fin de partida

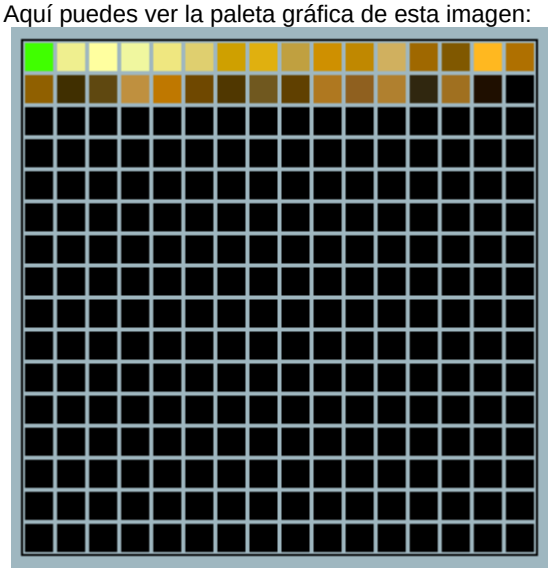
Al finalizar cada la partida, se presenta una imagen superpuesta en pantalla para indicar el fin de la partida.

Para modificar esta imagen, que por defecto es la de una calavera con 'Game Over', deberemos editar la zona que ocupa esta imagen en el siguiente fichero que encontraremos en la carpeta 'Bmp' de 'Entidad 3D':



*PrimerNivel*_hud_graficos.bmp

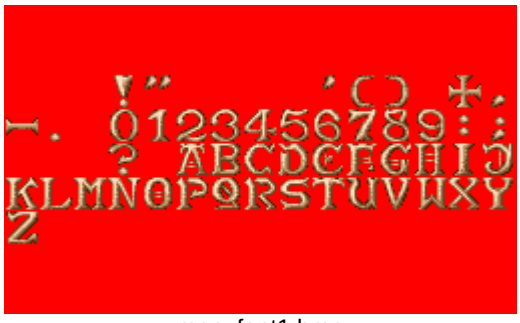
La imagen ha de ser un BMP con una paleta gráfica de 256 colores, siendo el color 0 (el primero de la paleta) el que indique el color transparente para la imagen. Aquí puedes ver la paleta gráfica de esta imagen:



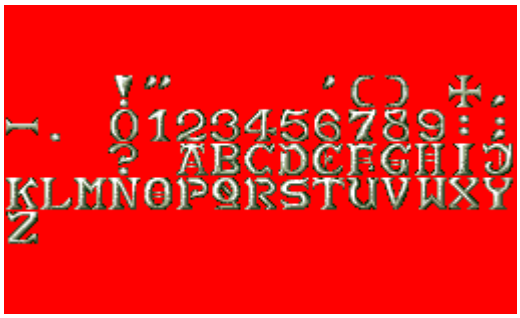
Imágenes del menú

Al pulsar la tecla 'Esc' aparece un menú superpuesto al escenario del juego. El tipo de letra, así como algunos elementos podremos modificarlos editando sus ficheros BMP.

Para modificar el tipo de letra deberemos editar las zonas que definen cada carácter en los siguientes ficheros que encontraremos en la carpeta 'Bmp' de 'Entidad 3D':

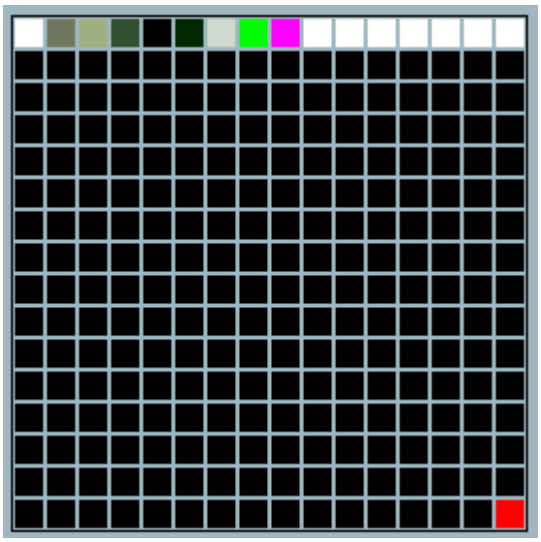


menufont1.bmp



menufont2.bmp

La imagen de 'menufont2.bmp' se utiliza para resaltar la selección que esté en ese momento en el menú.
Las imágenes han de ser BMPs de una resolución de 256x154 y con una paleta gráfica de 256 colores, siendo el color 255 (el último de la paleta) el que indique el color transparente para las imágenes. Aquí puedes ver la paleta gráfica de estas imágenes:



Desde este menú, el jugador también puede visualizar una pantalla que podrá mostrar los créditos de nuestro juego, lo único que debemos hacer es poner un fichero gráfico llamado '*PrimerNivel*_creditos.bmp' en la carpeta 'Bmp\menu'.

*PrimerNivel*_creditos.bmp

En realidad no sólo puedes utilizar ficheros gráficos .BMP para la pantalla de créditos, también puedes usar algún formato gráfico de los siguientes:
BMP, JPG, GIF, LBM, IFF, PBM, PCX, PGM, PNG, TGA, TIF, PSD, ICO

Imágenes de los textos

Los textos que aparecen en pantalla durante el juego lo definen ficheros BMP que podremos editar para modificar su aspecto.

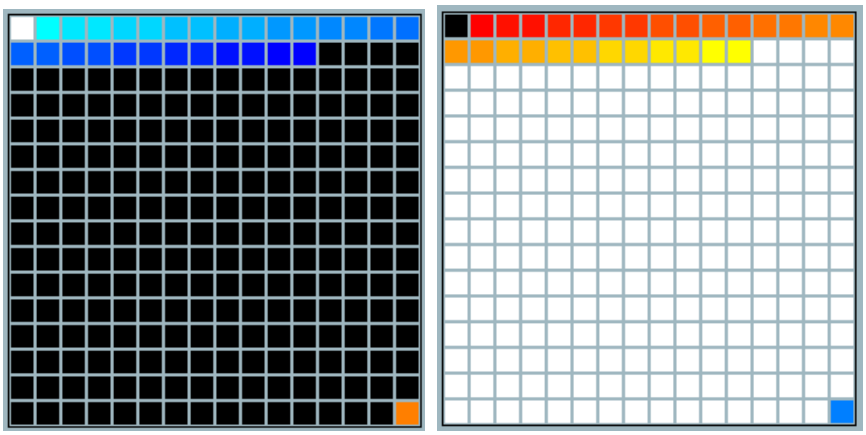
Para modificar el primer tipo de letra (utilizado para mensajes del sistema y el que utiliza el comando 'TEXT') deberemos editar las zonas que definen cada carácter en el siguiente fichero que encontraremos en la ruta de carpetas 'Bmp\Console\640x480' de 'Entidad 3D':

fuente.bmp

Mientras que si queremos modificar el segundo tipo de letra (el utilizado por el comando 'DIAL') deberemos editar el siguiente fichero que encontraremos en la ruta de carpetas 'Bmp\Console\640x480' de 'Entidad 3D':

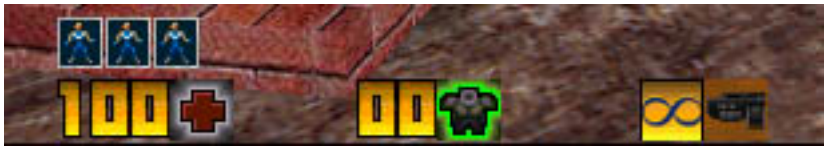
fuente2.bmp

Las imágenes han de ser BMPs de una resolución de 1024x24 y con una paleta gráfica de 256 colores, siendo el color 255 (el último de la paleta) el que indique el color transparente para las imágenes. Aquí puedes ver las paletas gráficas de estas imágenes:



Imágenes del marcador

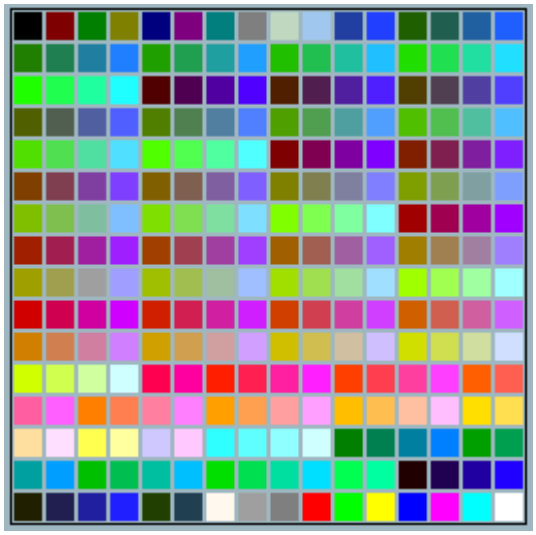
Los gráficos y números que definen el marcador podremos modificarlos editando dos ficheros BMP.



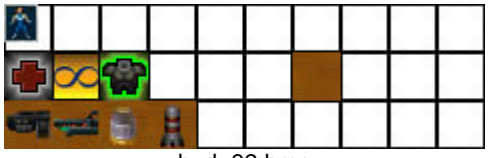
Para modificar los números que aparecen en el marcador deberemos editar las zonas que definen cada número en el siguiente fichero que encontraremos en la carpeta 'Bmp' de 'Entidad 3D':

hud_font2.bmp

La imagen ha de ser un BMP de una resolución de 200x44 y con una paleta gráfica de 256 colores, siendo el color 250 el que indique el color transparente. Aquí puedes ver la paleta gráfica de esta imagen:



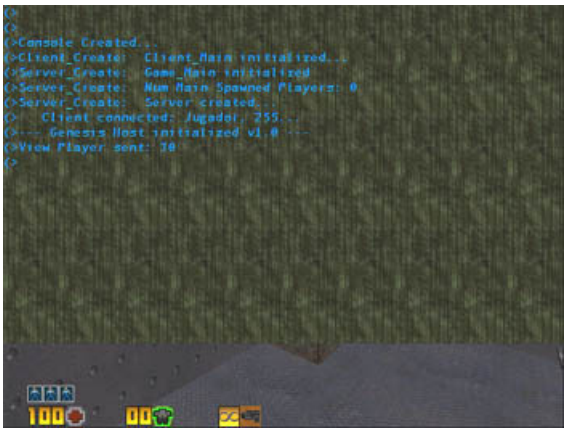
Mientras que si queremos modificar el resto de elementos que forman el marcador deberemos editar el siguiente fichero que encontraremos en la carpeta 'Bmp' de 'Entidad 3D':



hud_02.bmp

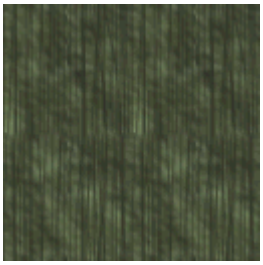
La imagen han de ser un BMP de una resolución de 240x72 y la paleta gráfica puede ser RGB. Esta imagen no contiene transparencias.

Imagen de la consola



La consola que podemos hacer aparecer con la tecla del 'Tabulador', tiene como fondo un fichero gráfico BMP que podremos modificar.

Para modificar este fondo deberemos editar el siguiente fichero que encontraremos en la ruta de carpetas 'Bmp\Console\640x480' de 'Entidad 3D':



consola.bmp

La consola sirve para que el jugador pueda entrar textos o comandos durante el juego.
También se utiliza para entrar ciertos comandos que le permiten al jugador hacer trampas en los juegos.

En la presente versión de Entidad 3D existen los siguientes comandos disponibles:
INMUNE (para conseguir inmunidad total)
ARMAS (todas las armas y munición infinita)
ENEMIGOS (enemigos no disparan)
VOLAR (activa/desactiva un modo que te permite volar atravesando todos los bloques del escenario como paredes, techos y suelos. En este modo, si atraviesas los límites del escenario el programa tendrá un mal funcionamiento y mostrará imágenes deformadas)
STATS (realiza un muestreo de FPS (Frames per second) 'Cuadros por segundo' durante 30 segundos, para medir el rendimiento del programa. El resultado lo presenta en pantalla y lo salva en el fichero 'informe_entidad_3d.log')
MEM (presenta la cantidad de memoria que está usando el programa. Para que este dato sea lo más fiable posible, usa este comando con el mínimo de programas corriendo al mismo tiempo en Windows).

» Consulta la entidad 'Configuracion' para saber como hacer que el jugador no pueda usar comandos de consola para hacer trampas en el juego.

Imagen de objetos del inventario

Para cada objeto que tengamos en nuestro juego tendremos que definirle una pequeña imagen de un tamaño de 32x32 que representará ese objeto en el inventario del jugador.

Inventario de objetos del jugador

Todas estas pequeñas imágenes de los objetos, están en un mismo fichero y han de colocarse de una manera muy concreta para que 'Entidad 3D' pueda tratarlas de forma correcta.
El orden de estas imágenes es de arriba a abajo, pegados a la izquierda y sin separación entre ellos. Por lo tanto el primer cuadro de 32x32 en la parte superior izquierda pertenecerá al objeto número 1, que es el objeto caja (1_caja.act) y debajo de este estaría la imagen del objeto 2, y así sucesivamente.
Aquí puedes ver un ejemplo con 3 objetos ya definidos y con bloques vacíos para más objetos:

*PrimerNivel*_objetos_distintos.bmp

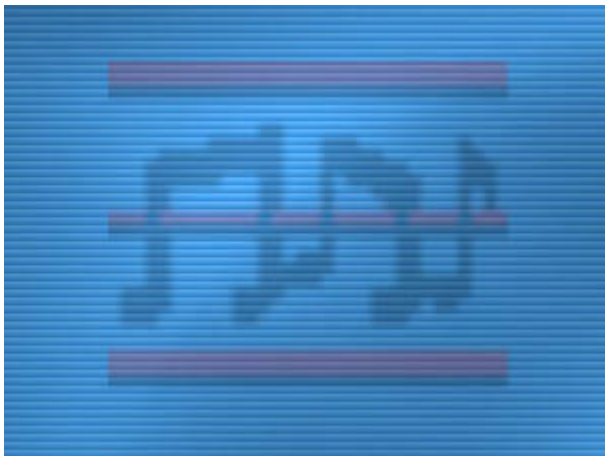
Esta imagen .BMP la tenemos que copiar en la carpeta 'Bmp' de 'Entidad 3D'.
La imagen ha de ser un BMP que puede ser de dimensiones variable si se respeta al menos un ancho de 64. La paleta gráfica puede ser RGB ya que esta imagen no soporta transparencias.

---

*» Para hacer aparecer/desaparecer el inventario de objetos en pantalla, el jugador debe usar la tecla **'i'**.*
*» Las imágenes que forman el panel que representa el inventario de objetos las encontrarás en: '**bmp\menu\inventario.bmp**' y '**bmp\menu\a_inventario.bmp**'*

Música de introducción

Si hemos utilizado la posibilidad de dotar a nuestro juego con una secuencia de imágenes para la introducción, es posible que queramos acompañar esa serie de imágenes con algún fondo musical.
Esta opción soporta ficheros de audio tipo: mp3, wav, mod, s3m, xm, it, mtm, mo3, mp1, mp2 y ogg.

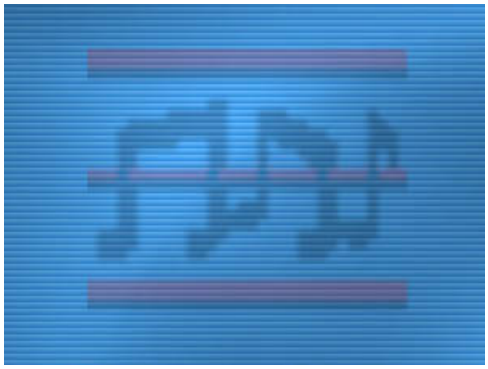


Para conseguir esto, es tan simple como copiar el fichero de audio que nos interesa con el nombre '*PrimerNivel*_intro.' en la carpeta 'Wav' de 'Entidad 3D'.
'PrimerNivel' será el nombre del primer nivel de que consta el juego.
La extensión de este fichero 'intro' será la que tenga el fichero de audio original que queremos interpretar.
Por ejemplo: Tenemos el fichero 'banda.mp3' que queremos poner como fondo musical para la introducción de nuestro juego que consta de los niveles (puente.bsp, carretera.bsp y escape.bsp). Pues copiamos el fichero MP3 con el nombre 'puente_intro.mp3' en la carpeta 'Wav', al dar comienzo la introducción con las imágenes, este será el fichero de audio que interpretará de fondo.
Si lo que tenemos es el fichero 'dark.mod', pues lo copiaremos con el nombre 'puente_intro.mod'.
Si no queremos poner ningún fondo musical para la introducción, pues no pondremos este fichero de audio en la carpeta 'Wav'.

Consulta la sección 'Imágenes de introducción' para ver como crear la secuencia de imágenes que acompañan al fondo musical.

Música de fondo

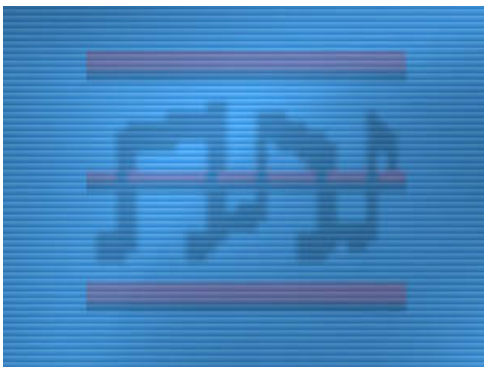
Durante el transcurso del juego podemos incluir una serie de fondos musicales continuos para acompañar el desarrollo de la aventura.
Esta opción soporta ficheros de audio tipo: mp3, wav, mod, s3m, xm, it, mtm, mo3, mp1, mp2 y ogg.



Pues lo único que hay que hacer para lograr esto, es copiar el fichero de audio que nos interesa con el nombre '*Nivel*_fondo.' en la carpeta 'Wav' de 'Entidad 3D'.
'Nivel' es el nombre del nivel al que queremos asignarle el fondo musical, ya que cada nivel podrá tener su fondo musical independiente.
La extensión de este fichero 'fondo' será la que tenga el fichero de audio original que queremos interpretar.
Por ejemplo: Tenemos el fichero 'newage.s3m' que queremos poner como fondo musical en el nivel 'hotel.bsp'. Pues copiamos el fichero 'newage.s3m' con el nombre 'hotel_fondo.s3m' en la carpeta 'Wav', al cargarse el nivel 'hotel.bsp' (ya sea el primer nivel del juego o no) este será el fichero de audio que se interpretará de fondo durante el transcurso de ese nivel.
Si lo que tenemos es el fichero 'dream.it', pues lo copiaremos con el nombre 'hotel_fondo.it'.
Si no queremos poner ningún fondo musical para un nivel en concreto, sólo tenemos que asegurarnos de que su fichero de 'fondo' no esté en la carpeta 'Wav'.

Consulta la sección del comando 'SONF' para ver como cambiar la música de fondo del juego durante el transcurso del mismo.

Música fin de partida

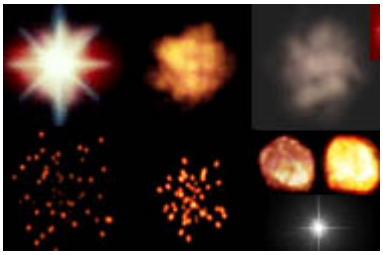


Al finalizar la partida del jugador (cuando ya se le han acabado todas las vidas) podemos interpretar una música que acompañará a la imagen de fin de partida del juego.
Esta opción soporta ficheros de audio tipo: mp3, wav, mod, s3m, xm, it, mtm, mo3, mp1, mp2 y ogg.

Para ello basta con copiar el fichero de audio que nos interesa con el nombre '*PrimerNivel*_final.' en la carpeta 'Wav' de 'Entidad 3D'.
'PrimerNivel' será el nombre del primer nivel de que consta el juego.
La extensión de este fichero 'final' será la que tenga el fichero de audio original que queremos interpretar.
Por ejemplo: Tenemos el fichero 'muerto.wav' que queremos poner como efecto de sonido al finalizar la partida de nuestro juego que consta de los niveles 'puente.bsp', 'carretera.bsp' y 'escape.bsp'.
Pues copiamos este fichero con el nombre 'puente_final.wav' en la carpeta 'Wav', al morir el jugador (después de agotar todas sus vidas), este será el fichero de audio que se interpretará, independientemente del nivel en el que muera el jugador.
Si lo que tenemos es el fichero 'death.xm', pues lo copiaremos con el nombre 'puente_final.xm'.
Si no queremos poner ninguna música o efecto de sonido al finalizar la partida, sólo tenemos que asegurarnos de que no esté este fichero de audio en la carpeta 'Wav'.

Otros ficheros

Existen otros ficheros BMP y WAV que también podrás modificar para adaptarlos a tu juego.



Estos ficheros definen entre otras cosas:
- Texturas del fuego de las armas.
- Texturas de las explosiones.
- Texturas que utilizan algunas entidades.
- Control de desplazamiento en menús.
- Diversos efectos de sonido.

Los ficheros BMP los encontrarás en la carpeta 'Bmp' o en carpetas que parten de esta.
Los ficheros WAV están en la carpeta 'Wav'. Estos ficheros .WAV deben ser archivos de sonido con la codificación normal estándar PCM de Windows. Te comento esto porque existen ficheros de sonido también con la extensión .WAV pero con otro tipo de codificación interna de audio. Si Entidad 3D intenta cargar uno de estos archivos de sonidos, el programa te presentará un error informándote del archivo .WAV que no puede cargar.

ANEXOS

Inventario de objetos y menú de acciones

Inventario de objetos del jugador (panel 'Inventario') con su panel de descripción encima
Menú de acciones (panel 'Acciones')

Inventario de objetos:

El inventario de objetos muestra al jugador todos los objetos de los que dispone, así como información adicional de cada uno de ellos a medida que los selecciona.

Si un objeto tiene un número de usos determinado, se indicará con unos números de color blanco en el interior del icono del objeto.

El jugador puede seleccionar los objetos desplazando el marco rojo usando las teclas derecha e izquierda del teclado numérico.

Si el jugador quiere que el inventario no tenga seleccionado ningún objeto, debe situar el marco rojo de selección en la posición superior izquierda (en el icono de la bola roja).

En el panel de descripción situado encima del inventario se presentan los siguientes datos del objeto seleccionado:
- Cuatro líneas de texto describiendo el objeto. Son las cuatro líneas introducidas en la entidad 'ItemObject' del objeto, parámetros 'Descrip1', 'Descrip2', 'Descrip3' y 'Descrip4'.
- Precio de venta del objeto por parte del jugador. Es el precio introducido en el parámetro '**Precio_venta**' de la entidad 'ItemObject' del objeto.

Menú de acciones:

El menú de acciones lo puede usar el jugador para seleccionar una acción a realizar combinada con el objeto que tenga seleccionado en el inventario de objetos. Esto permite crear juegos donde el uso concreto de objetos, de una manera determinada y en lugares y momentos específicos, resulta imprescindible en el desarrollo de la trama del juego.

El jugador puede seleccionar las acciones desplazando el marco rojo usando las teclas arriba y abajo del teclado numérico.

Para realizar la acción deseada el jugador finalmente debe pulsar la tecla 'Acción'

Si se desea realizar una acción determinada pero sin utilizar ningún objeto del inventario, el jugador debe seleccionar previamente el icono de la bola roja que tiene en su inventario (arriba a la izquierda).

Las imágenes que forman el panel que representa el inventario de objetos son:
bmp \ menu \ inventario.bmp
bmp \ menu \ a_inventario.bmp
bmp \ menu \ inventario_descripcion.bmp
bmp \ menu \ a_inventario_descripcion.bmp
Las imágenes que forman el panel que representa el menú de acciones son:
bmp \ menu \ acciones.bmp
bmp \ menu \ a_acciones.bmp

---

*» El jugador podrá hacer aparecer el inventario de objetos junto con el menú de acciones pulsando la tecla '**i**'.*
» Consulta la entidad 'Marcador' para saber como hacer invisibles el inventario y el menú de acciones, si tu juego no los utiliza.
» Consulta la sección de la entidad 'AccionMenu' para saber como se procesan las acciones que selecciona el jugador.

Funcionamiento de la puntuación

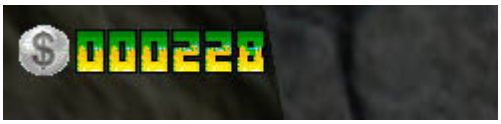
En Entidad 3D disponemos de la posibilidad de incluir puntuación en nuestro juego para ir incrementándola durante el transcurso del mismo en función de las acciones del jugador. Así mismo, de forma automática, Entidad 3D salva en un fichero la mayor puntuación alcanzada por el jugador para mostrarla también en pantalla.
Para modificar la puntuación del marcador durante el juego sólo tendremos que utilizar la variable 'PUNTUACION' de la misma manera que se utiliza cualquier variable de Entidad 3D mediante los comandos para tratar cifras.
Por ejemplo, si queremos aumentar en 100 puntos el marcador del jugador, utilizaríamos el siguiente comando:
ACCI OPE+ PUNTUACION 100
Podemos utilizar cualquier comando que se utiliza con variables con números para gestionar la puntuación, hasta podríamos quitar puntuación al jugador usando el comando 'OPE-'.
El control del marcador que muestra el récord es completamente automático y no tenemos que hacer nada. Este récord se guarda en la carpeta 'records' de Entidad 3D, por lo que si nos interesa borrarlo sólo tenemos que eliminar el fichero '.dat' que corresponda con el juego que queremos eliminar su récord. Estos ficheros '.dat' tienen el mismo nombre que el primer nivel del juego que le corresponde.
En cuanto a los gráficos que se utilizan para mostrar los títulos de los marcadores y los números que forman las puntuaciones, los podrás encontrar en el fichero 'hud_font2.bmp' dentro de la carpeta 'bmp' de Entidad 3D. Puedes editar este fichero para modificar estos gráficos, pero recuerda dejar ese color verde de fondo (color número 250) que hace de color transparente.

Porción del fichero 'hud_font2.bmp' utilizada para la puntuación

---

Consulta la entidad 'Marcador' para saber como ocultar o hacer aparecer los indicadores de puntuación y récord en tu juego.

Funcionamiento del dinero



En Entidad 3D tenemos la posibilidad de incluir el concepto de dinero en nuestro juego, y al igual que hacemos con la puntuación, podremos ir incrementándolo o reduciéndolo durante el transcurso del juego en función de las acciones del jugador.
La moneda que se utiliza para el dinero no es ninguna en concreto para que seas tú el que la definas en los mensajes que presentes al jugador o con algún nuevo diseño para el icono que acompaña a la cantidad de dinero.
Para modificar la cantidad de dinero durante el juego sólo tendremos que utilizar la variable '**DINERO**' de la misma manera que se utiliza cualquier variable de Entidad 3D mediante los comandos para tratar cifras.
Por ejemplo, si queremos aumentar en 55 el indicador de dinero, utilizaríamos el siguiente comando:
ACCI OPE+ DINERO 55
... o bien, si queremos restar 15, utilizaríamos el siguiente comando:
ACCI OPE- DINERO 15
Podemos utilizar cualquier comando que se utiliza con variables con números para gestionar el dinero, incluidos comandos 'IF COND'.
El control del indicador que muestra la cantidad de dinero es completamente automático y no tenemos que hacer nada. Este número se actualiza automáticamente cada vez que modificamos el valor de la variable '**DINERO**'.
En cuanto a los gráficos que se utilizan para mostrar el icono del dinero y los números que forman la cantidad de dinero, los podrás encontrar en el fichero '**hud_font2.bmp**' dentro de la carpeta '**bmp**' de Entidad 3D. Puedes editar este fichero para modificar estos gráficos, pero recuerda dejar ese color verde de fondo (color número 250) que hace de color transparente.

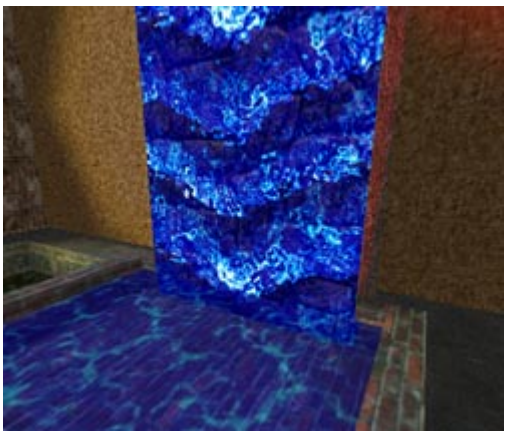
***»** Consulta la entidad 'Marcador' para saber como ocultar o hacer aparecer el indicador de dinero en tu juego.*

Funcionamiento de la opción 'Cargar partida' del menú de juego

Al comenzar cada nivel de juego en Entidad 3D se salva en el disco duro la posición que ocupa el jugador al dar comienzo el nivel, de manera que cuando vuelva a jugar otra partida al mismo juego disponga de la opción de comenzar esa partida a partir del último nivel que alcanzó. De esta forma si el jugador supera niveles nunca perderá el progreso que haya realizado.
El control de salvado de la partida es completamente automático y no tenemos que hacer nada para guardar las partidas. La partida se guarda en la carpeta '**partidas**' de Entidad 3D, por lo que si nos interesa borrarla sólo tenemos que eliminar el fichero '.dat' que corresponda con el juego que queremos eliminar su partida salvada. Los nombres de estos ficheros '.dat' comienzan con el nombre del primer nivel del juego que le corresponde.
Al cargar una partida también se cargará la puntuación y las vidas que tenía el jugador cuando se salvó la partida, es decir, justo al dar comienzo el nivel.

Entidad 3D no dispone de ninguna opción para que el jugador pueda salvar la partida en cualquier momento del juego.

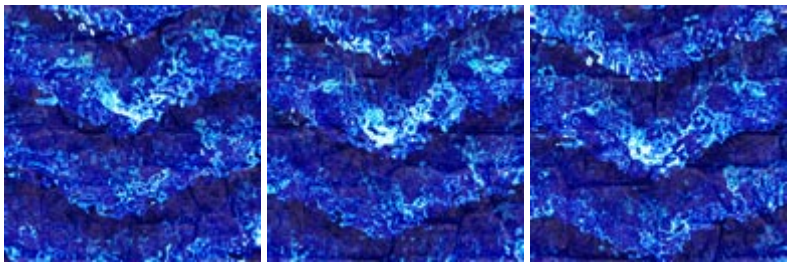
Texturas animadas



En nuestros escenarios creados con World Editor tenemos la posibilidad de poder asignar texturas animadas en cualquier parte del escenario, de esta forma podremos dotar de más vida a nuestros niveles.
Estas animaciones son en realidad secuencias de imágenes en formato .BMP que deberemos crear en su orden correcto para crear la animación.

Secuencias de imágenes:

La idea es sustituir una textura que hayamos utilizado en la construcción de nuestro nivel, por una secuencia de texturas para que de esta forma, durante el juego, se vean en la secuencia correcta para formar la animación.
Por ejemplo, para crear el efecto de una cascada habríamos creado los siguientes ficheros (**cascada1.bmp, cascada2.bmp y cascada3.bmp**):



Fichero .PRC:

Para realizar los cambios de texturas deberemos crear un fichero con extensión .PRC donde se indicarán diversos parámetros para sustituir las texturas. Ha de haber un fichero .PRC distinto por cada nivel al que queramos aplicar texturas animadas.
Por ejemplo, si tenemos un nivel que se llama 'calle.bsp' pues tendremos que crear un fichero llamado 'calle.prc'.
Estos ficheros .PRC deben estar en la misma carpeta que los niveles, en la carpeta 'levels' de Entidad 3D.
El contenido de este fichero para el ejemplo anterior sería:

procedurals
{
wbm: roca Animar(cascada, 3, 0.1, True);
}
Lo de 'procedurals' y los corchetes '{' y '}' nunca varía y las lineas que empiezan por 'wbm:' indicarán un cambio de texturas cada una.
En este caso lo que le estamos indicando a Entidad 3D con este fichero 'calle.prc' es que cuando estemos jugando al nivel 'calle' la textura llamada '**roca**' que utilizamos en World Editor como textura para lo que sea, la cambie por la secuencia '**cascada**', secuencia que la componen **3** imágenes, que se espere **0.1** segundos entre cada cambio de imagen, y con '**True**' le indicamos que puede hacer todo eso desde el mismo instante que comience el nivel.
Así pues esta es la sintaxis de las lineas 'wbm':
wbm: **textura_a_cambiar** Animar(**secuencia_bmp**, **numero_de_imágenes**, **retardo**, **activada**);
textura_a_cambiar: Nombre de la textura que se utilizó en Word Editor y que se desea sustituir por la animación.
secuencia_bmp: Nombre de la secuencia de imágenes .bmp que sustituirá a la textura que hemos indicado anteriormente.
Esta secuencia de imágenes siempre ha de comenzar con '1' y ser progresiva (Ejemplo: llave1.bmp, llave2.bmp, llave3.bmp, llave4.bmp, llave5.bmp, ...)
Todos los ficheros que componen la secuencia deben estar en la carpeta 'bmp\texturas' de Entidad 3D.
Deben ser imágenes de la misma resolución que la textura que se quiere sustituir.
Los tamaños permitidos de estas imágenes pueden ser: 32x32, 64x64, 128x128 o 256x256.
Las imágenes han de ser de 256 colores (de 0 a 255), pero el color 255 (el último) no se debe utilizar en las imágenes, de lo contrario se producirán extraños efectos en algunas zonas de la animación.
numero_de_imágenes: Número máximo de imágenes que componen la secuencia.
Indicar aquí el último número que hemos utilizado en la secuencia de imágenes .bmp que hemos creado.
retardo: Tiempo en segundos entre cada imagen de la secuencia.
Variando este dato podremos presentar la animación más rápida o más lenta.
activada: Poner 'True' si se desea que la textura animada comience a funcionar desde el mismo instante que empiece el nivel, de lo contrario poner 'False'.
Durante el transcurso del juego podremos activar o desactivar texturas animadas utilizando los comandos 'TEA1' y 'TEA0'.

Pero no sólo puedes poner una línea de cambio de texturas por nivel, sino que puedes poner las que quieras, todos los cambios de texturas podrán funcionar, sin problemas y con suavidad, simultáneamente.
Aquí puedes ver el contenido de un fichero .PRC con 4 cambios de texturas:
procedurals
{
wbm: roca Animar(cascada, 3, 0.1, True);
wbm: brick Animar(plasma, 5, 0.5, False);
wbm: monitor Animar(alarma, 2, 3, False);
wbm: agua Animar(aguasec, 6, 0.2, True);
}

¿ Para que más sirven las texturas animadas ?:
Fluidos más realistas (lava, agua, residuos, barro, cascadas).
Presentar animaciones en pantallas y monitores.
Crear habitaciones psicodélicas.
Publicidad dinámica en paredes de calles o estadios.
Letreros luminosos.
Objetos del escenario con texturas cambiantes.
Paredes vivientes tipo 'Silent Hill 3'.

Las cámaras en Entidad 3D

Cuando nos planteamos realizar un juego, una de las primeras cosas en las que pensamos es de que manera presentaremos al jugador el escenario o pantallas de que constará nuestro juego. Esta elección determina en muchas ocasiones el tipo de juego que al final nos quedará: FPS (primera persona), juego de disparos en tercera persona, aventura 3D en primera persona, aventura 3D en tercera persona, RPG (vista aérea), plataformas, arcade de una pantalla, etc..., pero en ningún lugar de Entidad 3D encontrarás una selección de tipo de juego a realizar, aunque si el tipo o tipos de cámaras que vas a utilizar.
Entidad 3D nació con la intención de poder realizar juegos en primera persona o como máximo en tercera persona, ya fueran de disparos o no, pero con el tiempo y las diversas modificaciones que he implementado al programa, sobre todo a nivel de cámaras, he podido observar que, sin hacer nada más en especial y con todas las características de las que ya disponía Entidad 3D, se podían hacer otros tipos de juegos que en principio no estaban previstos, como juegos de plataformas con vista de cámara sin rotación o vistas desde arriba sobre un escenario fijo donde transcurre toda la acción de un juego arcade al estilo de los juegos de máquinas recreativas de los años 80, pero ahora con la potencia 3D y sin sprites.
Con esto quiero decir que con Entidad 3D no sólo se pueden realizar los clásicos juegos en primera y tercera persona (sobre todo de disparos) sino que dispone de suficiente variedad en cuanto a cámaras, control de juego, comandos y entidades como para experimentar en una amplia gama de tipos de juego, evidentemente sin aspirar al nivel técnico de lo juegos comerciales actuales.
A continuación se presenta un resumen de las diversas cámaras que soporta Entidad 3D, ejemplos con imágenes de estas cámaras, así como imágenes de esos mismos tipos de cámaras en juegos comerciales.

Nombre de la cámara o vista: **Primera persona**

¿ Cómo se ve ?: En ocasiones también llamada cámara subjetiva, esta cámara presenta la escena como si el jugador lo viera con sus propios ojos. El jugador no ve al personaje protagonista que controla.

Variaciones de esta cámara: Si se desea, esta vista puede presentar al mismo tiempo el arma que esté utilizando el jugador.

¿ Para qué sirve ?: Para realizar los llamados juegos en primera persona. Pueden ser juegos de disparos FPS (First Person Shooter) o aventuras 3D.

¿ Cómo la puedo utilizar en mi juego ?: Debes utilizar la entidad 'Configuracion', consulta la sección de esta entidad. Para que se vea el arma que utiliza el jugador debes utilizar la entidad 'VerArma' y activar el parámetro 'Ver_arma' de la entidad 'Configuracion'.

¿ Puede el jugador activar esta cámara pulsando una tecla ?: Si, con la tecla 'Q'.

Ejemplos de esta cámara en Entidad 3D:

Ejemplos de esta cámara en juegos comerciales:

Far Cry

Quake II

Medal of honor

Serious Sam

Nombre de la cámara o vista: **Tercera persona**

¿ Cómo se ve ?: Es una cámara que se sitúa siempre detrás del jugador a cierta distancia y ligeramente elevada , por lo que esta vista siempre presenta la escena 3D con el jugador de espaldas. Esta cámara sigue continuamente al jugador y apuntando hacia él.
Tiene la propiedad de evitar chocar con bloques del escenario, variando de forma automática su posición para evitar estas colisiones, pero sin dejar de enfocar al jugador.

Variaciones de esta cámara: No hay.

¿ Para qué sirve ?: Para realizar los llamados juegos en tercera persona. Pueden ser juegos de disparos o aventuras 3D.

¿ Cómo la puedo utilizar en mi juego ?: Debes utilizar la entidad 'Configuracion', consulta la sección de esta entidad.

¿ Puede el jugador activar esta cámara pulsando una tecla ?: Si, con la tecla 'A'.

Ejemplos de esta cámara en Entidad 3D:

Ejemplos de esta cámara en juegos comerciales:

Tomb Raider

Max Payne 2

Manhunt

Red Dead Revolver

Nombre de la cámara o vista: **RPG**

¿ Cómo se ve ?: Es una vista aérea del jugador y del escenario pero ligeramente detrás de él. Se llama RPG por ser la vista más utilizada en este tipo de juegos (Role Play Game), pero eso no quiere decir que tengas que hacer un juego RPG para utilizar esta vista.
Esta cámara sigue continuamente al jugador y apuntando hacia él. Si el jugador gira, la cámara también gira para de esta forma siempre situarse detrás del jugador.

Variaciones de esta cámara: No hay.

¿ Para qué sirve ?: Para realizar juegos tipo RPG o juegos que necesitan presentar una amplia zona desde arriba y alrededor del personaje sin importar que los personajes del juego (incluido el del jugador) se vean más pequeños que con otras vistas.

¿ Cómo la puedo utilizar en mi juego ?: Debes utilizar la entidad 'Configuracion', consulta la sección de esta entidad.

¿ Puede el jugador activar esta cámara pulsando una tecla ?: Si, con la tecla 'Z'.

Ejemplos de esta cámara en Entidad 3D:

Ejemplos de esta cámara en juegos comerciales:

Pool of Radiance

Pool of Radiance

Tomb Raider

RPG

Nombre de la cámara o vista: **Fija**

¿ Cómo se ve ?: Presenta una vista del escenario en función de donde la hayamos situado y hacia donde la hayamos enfocado. En ninguna circunstancia estas cámaras se desplazarán por el escenario, aunque si pueden configurarse para que siempre enfoquen al jugador desde la posición donde se encuentren.
En un mismo escenario puedes situar diferentes cámaras de este tipo activándolas durante el transcurso del juego o al dar comienzo este.

Variaciones de esta cámara: Existe la opción de que enfoquen siempre al jugador o no.

¿ Para qué sirve ?: Para mostrar alguna parte del escenario durante un instante, estando el jugador en el cuadro de esa cámara o no. También se puede utilizar para realizar un juego completo con una sola de estas cámaras, donde todo el desarrollo del mismo (o de un nivel) se puede seguir a cierta altura y enfocando hacia un sólo lugar.

¿ Cómo la puedo utilizar en mi juego ?: Debes utilizar la entidad 'CamaraFija', consulta la sección de esta entidad.

¿ Puede el jugador activar esta cámara pulsando una tecla ?: No.

Ejemplos de esta cámara en Entidad 3D:

Ejemplos de esta cámara en juegos comerciales:

Bomberman 64

Elevator Action

Go Go Mile Smile

Max Payne

Nombre de la cámara o vista: **Plataformas**

¿ Cómo se ve ?: Se llama Plataformas por ser un tipo de cámara utilizada en algunos juegos muy populares de este tipo, pero al igual que la cámara RPG, no es obligatorio hacer un juego de plataformas con esta cámara, de hecho se pueden hacer juegos de plataformas con una cámara en tercera persona o con cámaras fijas.
Esta cámara presenta una vista del escenario y del jugador similar a la vista en tercera persona, pero esta cámara no gira buscando la espalda del jugador, por lo que este se podrá ver desde distintos ángulos según esté mirando hacia la cámara, de lado o de espaldas.
Esta cámara sigue al jugador manteniendo una distancia y altura fijas (aunque configurables) pero sin realizar ningún tipo de rotación, en realidad siempre enfoca en el mismo sentido pero buscando al jugador.

Variaciones de esta cámara: Se puede variar su altura y distancia con respecto al jugador, aunque esos valores permanecerán fijos durante todo el juego.

¿ Para qué sirve ?: Al ser configurable, nos permite obtener vistas que de otra manera no conseguiríamos con el resto de cámaras.
Permite realizar juegos 3D con un personaje que se ve desde cualquier ángulo y sin marear al jugador en rotaciones continuas de cámara
En combinación con el parámetro '**Avance_direccion**' puesto '**True**' (de la entidad 'Configuracion') permitirá controlar al personaje con sólo cuatro teclas (o con 4 botones del Joystick), consiguiendo de esta forma juegos 3D muy rápidos, adictos y fáciles de controlar por parte del jugador.

¿ Cómo la puedo utilizar en mi juego ?: Debes utilizar la entidad 'Configuracion', consulta la sección de esta entidad.

¿ Puede el jugador activar esta cámara pulsando una tecla ?: Si, con la tecla 'X'.

Ejemplos de esta cámara en Entidad 3D:

Ejemplos de esta cámara en juegos comerciales:

Bomberman Hero	Chamaleon Twist
Crash Bandicot	Rayman

Combina estas cámaras con el tipo de control del personaje protagonista para conseguir el sistema de juego que deseas.
Recuerda que puedes variar el tipo de control del personaje mediante el parámetro 'Avance_direccion' de la entidad 'Configuracion'

Encriptación de datos

(Como proteger tu juego para que no sea alterado)

Este apartado trata sobre la posibilidad que ofrece Entidad 3D para encriptar ciertos datos de tu juego para que no puedan ser vistos ni manipulados desde fuera del juego.
También habla sobre algunas medidas que puedes tomar para dificultar, en la medida de lo posible, que el jugador pueda hacer trampas en tu juego.

Puntos vulnerables a tener en cuenta y sus soluciones:

PUNTO 1:
Los ficheros de comandos **.INI** y los ficheros de diálogos **.DLG** que has creado en la carpeta '**comandos**', son completamente visibles y modificables por el jugador.
Hack:
El jugador puede aprovechar esto para cambiar o eliminar comandos de los ficheros **.INI** para alterar el comportamiento original de tu juego.
Mirando los **.DLG** o los propios **.INI** puede ver las respuestas que tiene que dar en el juego o los diálogos de todos los personajes.
Solución:
Convierte todos tus ficheros **.INI** y **.DLG** de tu juego que estén en la carpeta '**comandos**' a ficheros encriptados **.INI.ENC** y **.DLG.ENC**.
Evidentemente esto lo debes realizar cuando vayas a distribuir tu juego y cuando creas que tus ficheros .INI y .DLG ya no sufrirán más modificaciones. De todas formas el método de crear ficheros .ENC no es destructivo ya que no altera para nada los .INI o .DLG originales sino que crea nuevos ficheros .ENC encriptados.
Para crear estos ficheros .ENC utilizaremos el propio Entidad 3D. Deberás realizar estos pasos para crearlos:

Borra todo el contenido de la carpeta '**encriptados**' de Entidad 3D
Copia todos los ficheros .INI y .DLG que usa tu juego de la carpeta '**comandos**' a la carpeta '**encriptados**'.
Ejecuta tu juego de manera normal.
Sal de tu juego de manera normal.
Mira el contenido de la carpeta '**encriptados**', verás que han aparecido tantos ficheros .ENC como ficheros habías copiado. Estos ficheros son tus ficheros . INI y .DLG originales pero en su versión encriptada, si abres alguno de ellos podrás observar que no se entiende absolutamente nada.
Todos los ficheros .ENC que se han creado en la carpeta '**encriptados**' cópialos a la carpeta '**comandos**'.

Como ves, en ningún momento del proceso de creación de los ficheros encriptados, vas a perder el contenido original de tus ficheros .INI o .DLG.

A continuación borra todo el contenido de la carpeta '**encriptados**'
¿ Ya está ?, pues si ejecutas tu juego en este punto el programa seguirá usando los ficheros .INI y .DLG originales sin encriptar, aunque en la carpeta '**comandos**' estén sus equivalentes .ENC encriptados. Entonces .. ¿ cómo se hace para que el juego sólo use los ficheros .ENC ?. Pues muy sencillo: en la entidad 'Configuracion' (que seguramente habrás añadido en el primer nivel de tu juego) pon a 'True' el parámetro '**UsarFicherosEnc**'.
No tienes más que recompilar tu nivel y volver a ejecutar tu juego, en ese momento tu juego solamente usará los fichero .ENC. Haz la prueba moviendo los ficheros .INI y .DLG que usa tu juego a otra carpeta provisional, y digo mover porque no se te ocurra borrar los ficheros para hacer esta prueba.
Eso si, en la copia de la versión definitiva que prepares para la distribución de tu juego, borra todos los .INI y .DLG de la carpeta '**comandos**' , ya que el juego ya no los necesitará para nada al disponer de los ficheros encriptados .ENC

PUNTO 2:
El jugador puede modificar tu fichero 'autonivel.ini' o el fichero **.BAT** que da comienzo a tu juego empezando por el primer nivel.
Hack:
El jugador puede usar esto para cambiar el nombre del fichero .BSP que se lanza para evitar entrar en el juego por el primer nivel y de esta forma jugar al resto niveles sin seguir el orden correcto.
Solución:
Utiliza los comandos 'SETCLAVEJUEGO' y 'CHKCLAVEJUEGO' en ficheros de comandos .INI de tu juego.

PUNTO 3:
El jugador puede modificar los ficheros '**.dat**' de partidas salvadas que se guardan en la carpeta '**partidas**'.
Hack:
El jugador puede alterar estos ficheros para ponerse más vidas y más dinero, que conseguiría posteriormente utilizando la opción de cargar partida dentro del juego.
Solución:
En realidad no debes hacer nada para tener un mínimo de protección para estos ficheros, ya que el programa chequea la integridad de estos ficheros cada vez que se usa la opción de cargar partida, aunque para una mayor seguridad se recomienda que en tu juego uses el comando que se cita en el punto 2 (SETCLAVEJUEGO) ya que si el programa dispone de una clave de juego aumentará la seguridad de estos ficheros '**.dat**' de partidas salvadas.

PUNTO 4:
El jugador puede abrir la consola del juego y usar los comandos que permiten hacer trampas.
Hack:
Estos comandos permiten hacer inmune al jugador, darle todas las armas, munición infinita e incluso hacer que los enemigos no le disparen.
Solución:
En la entidad 'Configuracion' (que seguramente habrás añadido en el primer nivel de tu juego) pon a 'True' el parámetro '**UsarFicherosEnc**', aunque seguramente ya lo tendrás así si tu juego usa los ficheros encriptados **.enc**

PUNTO 5:
Los ficheros .BSP que forman los niveles de tu juego contienen datos que podrían ser manipulados.
Hack:
El jugador podría modificar estos ficheros .BSP para asignarse más dinero, vidas o alterar los nombres de los ficheros .INI que se utilizan en el nivel.
Solución:
Crea unos ficheros de control '**.bsp.chk**' por cada fichero '**.BSP**' que tenga tu nivel. Cuando tu juego esté funcionando con el parámetro '**UsarFicherosEnc**' a 'True' de la entidad 'Configuracion', se comprobarán unos códigos que contiene el fichero **'.bsp.chk'** con otros que contiene el fichero .BSP. En caso de que no coincidan, el programa dejará de funcionar.
Para crear estos ficheros **.CHK** deberás realizar estos pasos:
Borra todo el contenido de la carpeta '**encriptados**'
Copia todos los ficheros de nivel .BSP que utilice tu juego, de la carpeta '**levels**' a la carpeta '**encriptados**'.
Ejecuta tu juego pero añadiendo el parámetro **-clavejuego clave**, por ejemplo con esta línea:
Entidad3d.exe -name Jugador -map tunivel -PickMode -clavejuego Nemesis
Eso para crear los ficheros '**.chk**' usando por ejemplo la clave de juego **Nemesis**.
Esta misma clave es la que debes utilizar en los comandos 'SETCLAVEJUEGO' y 'CHKCLAVEJUEGO' que habrás añadido en tus ficheros de comandos .INI
Pulsa '**Cancelar**' en la selección del 'driver' de pantalla para salir del programa.
Mira el contenido de la carpeta '**encriptados**', verás que han aparecido tantos ficheros '**.bsp.chk**' como ficheros .BSP copiaste.
Todos los ficheros '**.bsp.chk**' que se han creado en la carpeta '**encriptados**' cópialos a la carpeta '**levels**'.
A continuación borra todo el contenido de la carpeta '**encriptados**'
A partir de este momento cada vez que ejecutes tu juego con '**UsarFicherosEnc**' a 'True', el programa buscará ficheros '**.bsp.chk**' en la carpeta '**levels**' para verificar el contenido de los ficheros de nivel .BSP.
*Nota: El parámetro **-clavejuego** sólo es necesario a la hora de crear los ficheros '**.bsp.chk**', no para lanzar tu juego de forma normal, ya sea una versión protegida o no.*

***»** Todos estos procesos no garantizan al 100% que tu juego no sea desprotegido usando diversas técnicas de 'cracking' o explotando vulnerabilidades en el sistema.*
De todas formas piensa que estos procesos ofrecen un mínimo de complejidad frente a la mayoría de la gente que pueda estar interesada en estudiar la manera de destripar tu juego para hacer trampas.

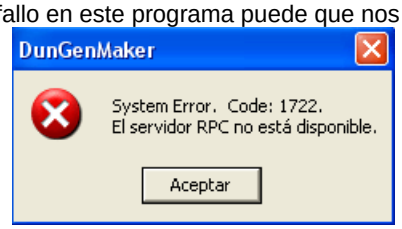
GENERADOR DE MASMORRAS PARA ENTIDAD 3D

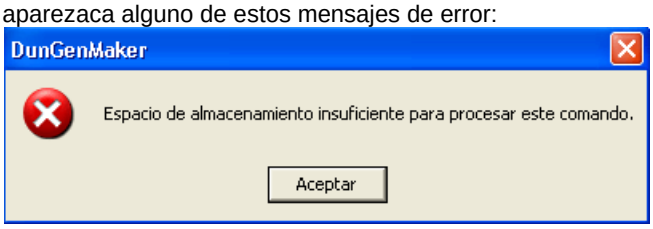
En este apartado veremos como generar mazmorras al azar usando Entidad 3D.
Esto nos permitirá crear escenarios de forma instantánea y automática para nuestros juegos, sin necesidad de saber crear mapas ni modelar.
Debido a la cantidad y variedad de texturas y modelos 3D .ACT que se usan en el proceso de generación de estas mazmorras, es prácticamente imposible que veamos dos mazmorras iguales por mucho que generemos mazmorras. Como es lógico el estilo de las mazmorras será similar, ya que siempre se usan las mismas texturas y los mismos modelos 3D .ACT, pero al ser distribuidos al azar nunca aparecerán combinados de la misma manera.
Para crear una de estas mazmorras básicamente lo realizaremos siguiendo estos cuatro pasos:
(A) Generación al azar de la distrubución de pasillos y habitaciones de nuestra mazmorra.
(B) Establecer algunos parámetros que nos permitirán variar diversas características de la decoración de la mazmorra.
(C) Usar una opción especial de Entidad 3D para levantar la mazmorra en 3D. Este proceso nos creará un fichero .MAP que cargaremos con World Editor.
(D) Cargar en World Editor el fichero .MAP generado anteriormente para compilarlo y transformarlo en .3DT y .BSP.

(A) Distribución de pasillos y habitaciones:
Esto lo realizaremos con el programa **Dungeon Maker** (de Peter Tyjewski) que incluye Entidad 3D.
Los pasos a seguir son los siguientes:

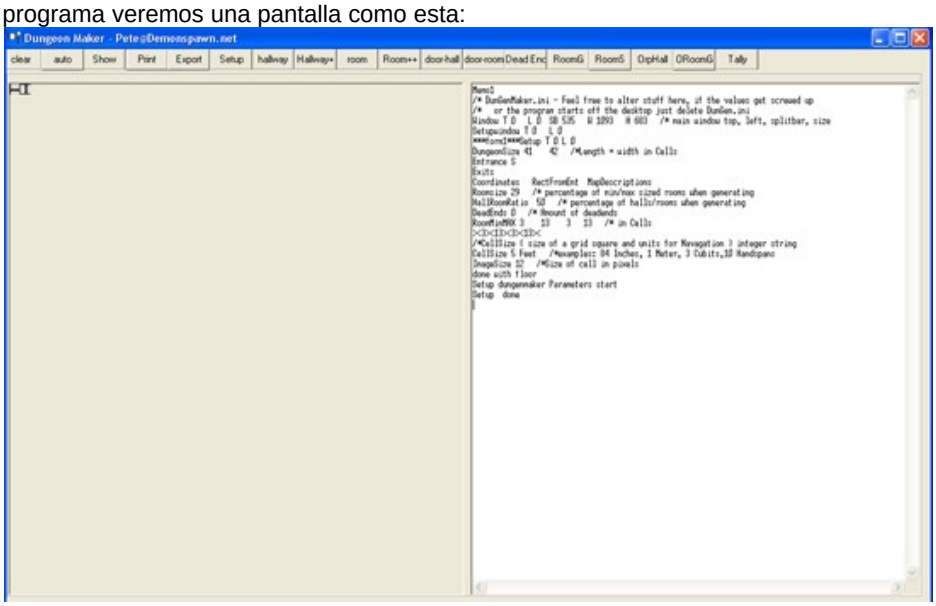
1.- Ejecutamos el programa usando el icono '**Dungeon Maker**'

que se encuentra en nuestra carpeta de Entidad 3D.
Debido a un fallo en este programa puede que nos aparezaca alguno de estos mensajes de error:



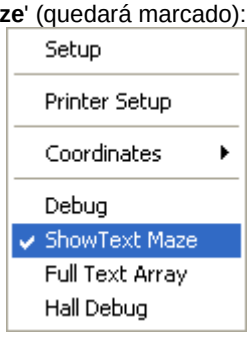


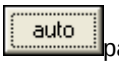
... o algún otro mensaje más esotérico.
Con pulsar '**Aceptar**' bastará para hacer desaparecer el mensaje para poder continuar. Si aparece en más ocasiones realizaremos lo mismo.
Una vez iniciado el programa veremos una pantalla como esta:



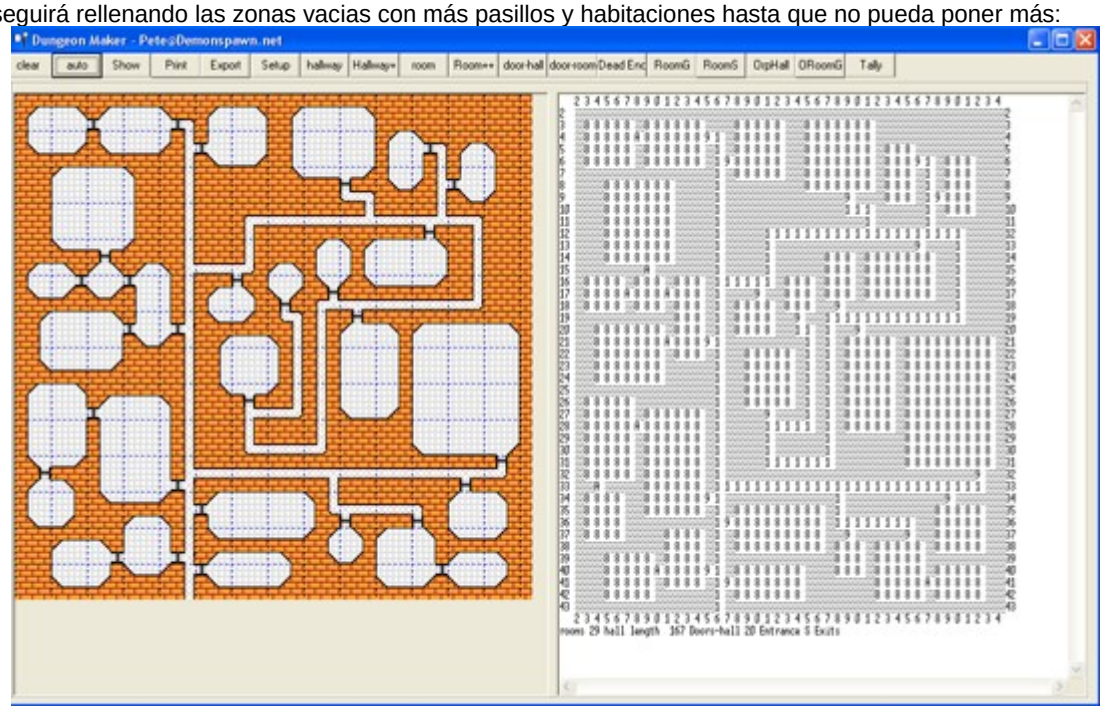
2.- Pulsamos el botón derecho del ratón sobre cualquier parte en la ventana del programa para hacer aparecer el siguiente menú:

Seleccionamos '**ShowText Maze**' (quedará marcado):

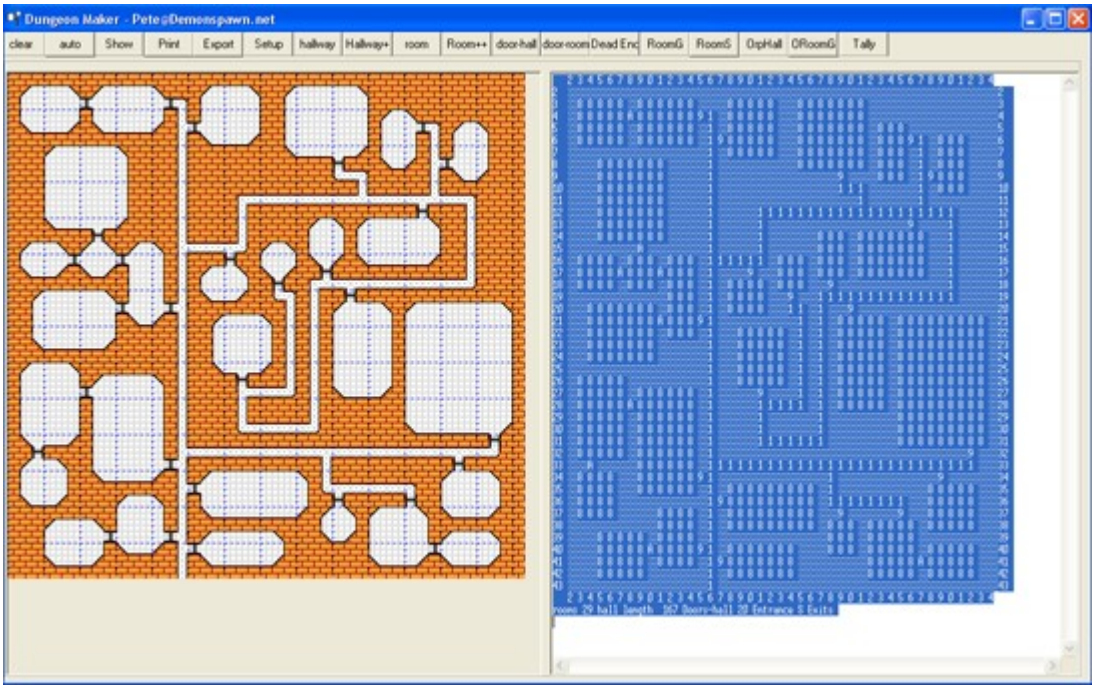


3.- Pulsamos el botón 

para generar la distribución de pasillos y habitaciones. Si volvemos a pulsar este botón, el programa seguirá rellenando las zonas vacias con más pasillos y habitaciones hasta que no pueda poner más:

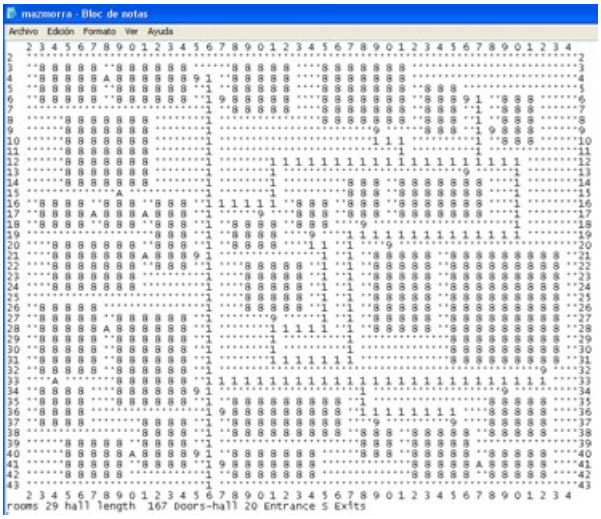


4.- Usamos el ratón (botón izquierdo y arrastrando) para seleccionar todo el bloque de texto y números que nos aparece en la parte derecha.



... a continuación copiamos todo lo seleccionado al portapapeles pulsando simultánemente las teclas **CTRL** y **C**.

5.- Abrimos el programa de block de notas de Windows o cualquier programa para la edición de textos simples. Usando este editor, abrimos el fichero '**dungeon.txt**' que se encuentra en la carpeta '**DungeonMaker**' de Entidad 3D. A continuación borramos todo su contenido y pegamos el contenido que tenemos en el portapapeles pulsando simultáneamente las teclas **CTRL** y **V**.
Finalmente salvamos el fichero:



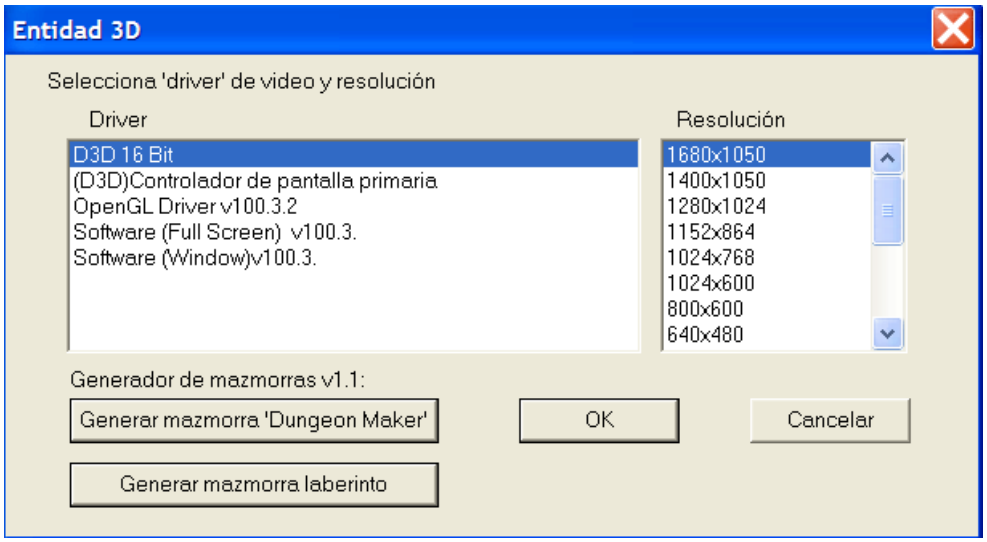
Si no te gusta la mazmorra que se ha generado y quieres generar otra diferente, sólo tienes que pulsar el botón clear y volver al punto **3.-**.

Antes de usar el botón auto para generar una nueva mazmorra, disponemos de una serie de opciones para variar algunos parámetros que afectarán a la generación de las mazmorras. Sólo tienes que pulsar el botón Setup para hacer aparecer estas opciones. Nos permitirán variar el tamaño de la mazmorra, la entrada, la salida, el tamaño máximo y mínimo de las habitaciones, la proporción de habitaciones y pasillos, etc...

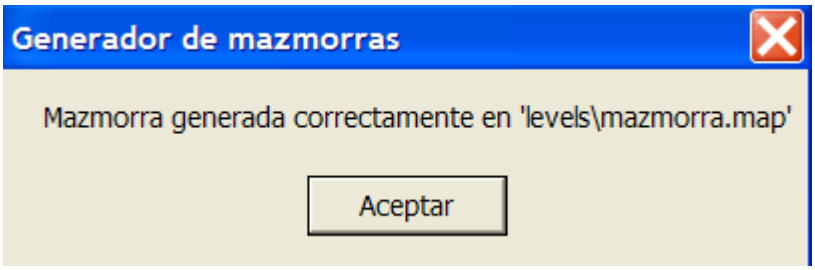
Si lo hemos realizado todo correctamente, en este punto tendremos el fichero '**dungeon.txt**' en la carpeta '**DungeonMaker**' que contendrá la distribución de pasillos y habitaciones de la mazmorra que necesita Entidad 3D para transformarla en 3D.

(B) Parámetros para variar la decoración de la mazmorra:
Estos parámetros los podrás establecer editando el fichero '**mazmorra.ini**' que encontrarás en la carpeta principal de Entidad 3D.
En el interior del mismo fichero encontrarás los nombres de los parámetros y los valores que pueden tener, así como algunos ejemplos y una explicación de su utilidad.
Es importante que por lo menos tengas puesto el siguiente parámetro como se indica a continuación, de lo contrario no te aparecerán los botones de generación de mazmorras y laberintos:
ver_botones_mazmorra = si

(C) Generación del fichero .MAP que contendrá la mazmorra:
Lanzamos cualquier nivel de Entidad 3D que nos pida seleccionar la resolución de pantalla, por ejemplo uno de los niveles que incluye Entidad 3D (Tiro al pato o Secta).
Si hemos establecido '**ver_botones_mazmorra = si**' en el fichero '**mazmorra.ini**', nos aparecerá la pantalla habitual de selección de resolución de pantalla y además dos botones más ('**Generar mazmorra 'Dungeon Maker**' y '**Generar mazmorra laberinto**'):

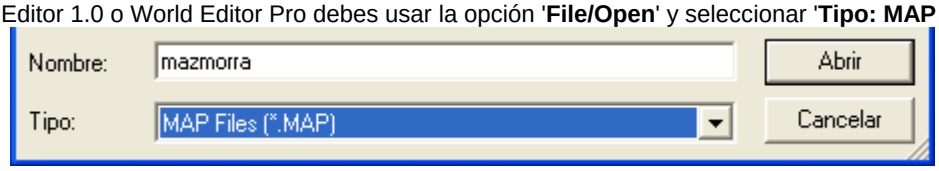


De momento pulsamos en la opción '**Generar mazmorra 'Dungeon Maker**'. Si todo ha ido bien nos aparecerá el siguiente mensaje:

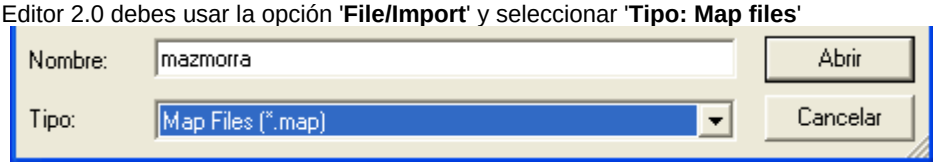


... que nos indicará que ya está disponible (en la carpeta '**levels**') el fichero .MAP de nuestra mazmorra para cargarlo con World Editor.

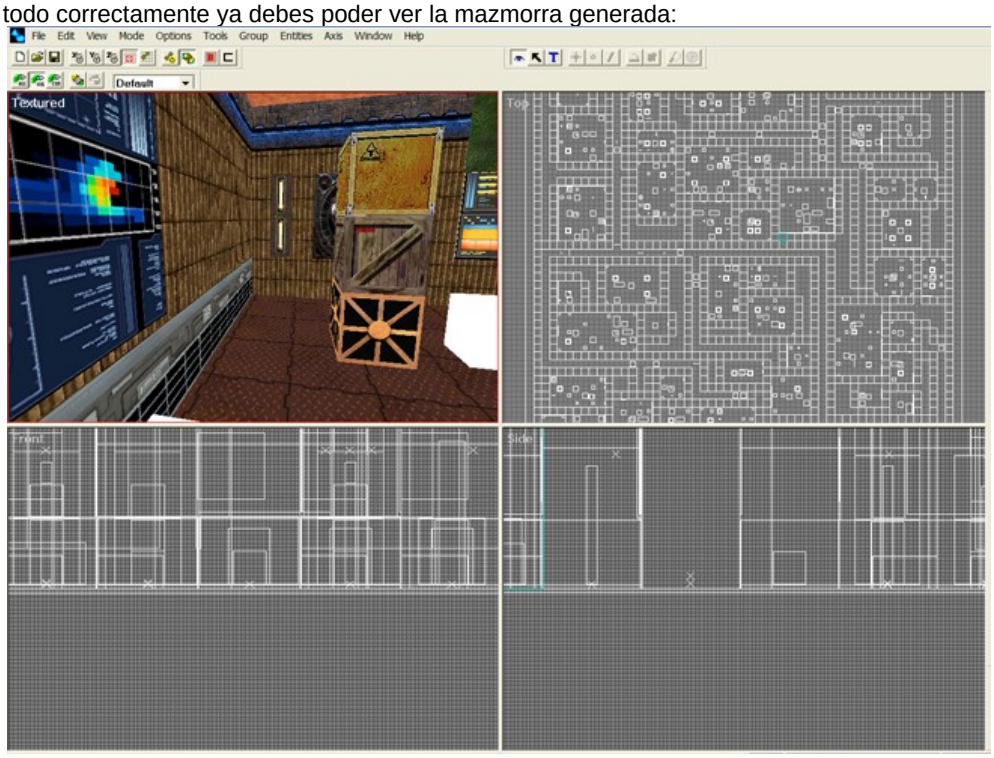
(D) Cargar en World Editor el fichero .MAP generado para compilarlo y transformarlo en .3DT y .BSP:
1.- Ejecutamos World Editor, ya sea la versión 1.0, 2.0 o Pro.
2.- Abrimos el nivel '**mazmorra_base.3dt**' que encontaremos en la carpeta '**levels**. Este nivel es un nivel vacio pero que establece el fichero de texturas que utilizará la mazmorra, además de servirnos de base para importar la mazmorra en el siguiente paso.
3.- Realizamos la importación del fichero .MAP que hemos generado en la sección **(C)**. Esta importación es ligeramente diferente en función de la versión de World Editor que se utiliza:
3a.- Si usas World Editor 1.0 o World Editor Pro debes usar la opción '**File/Open**' y seleccionar '**Tipo: MAP files**'



... y '**Abrir**' el fichero '**mazmorra.map**' que se encuentra en la carpeta '**levels**'.
3b.- Si usas World Editor 2.0 debes usar la opción '**File/Import**' y seleccionar '**Tipo: Map files**'

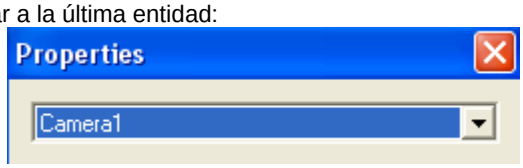


... y '**Abrir**' el fichero '**mazmorra.map**' que se encuentra en la carpeta '**levels**'.
Si has realizado todo correctamente ya debes poder ver la mazmorra generada:



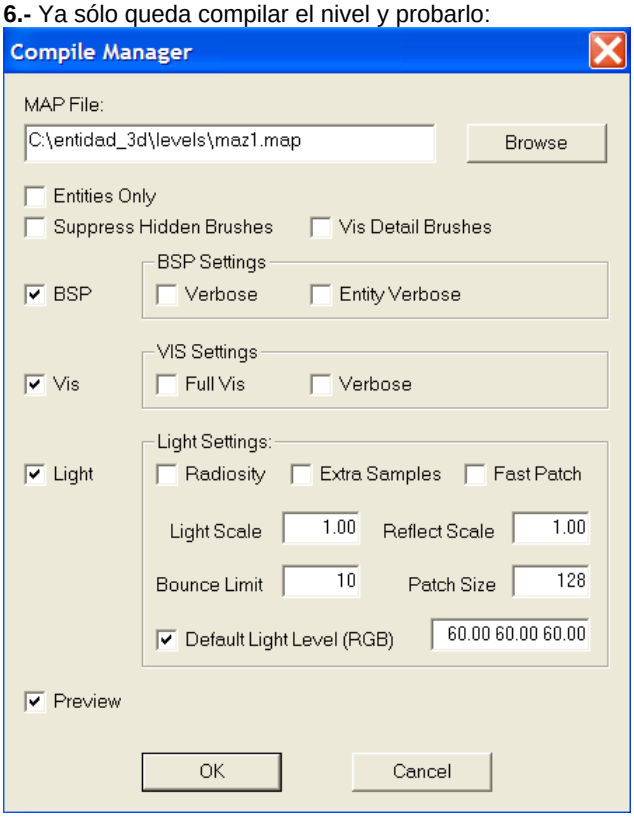
4.- Salvamos el nivel con la opción '**File/Save as**', por ejemplo con el nombre '**maz1**'. De esta forma evitaremos salvarlo por error con el nombre '**mazmorra_base**' que es el nivel base que debemos dejar siempre intacto.
Esto nos habrá creado un fichero normal de nivel '**maz1.3dt**' que podremos cargar con World Editor siempre que queramos para poderlo editar.

5.- Ahora tenemos que realizar la actualización de todas las entidades que contiene el nivel. Esto se realiza, en World Editor 1.0 y 2.0, seleccionando propiedades de las entidades, situándose en la primera entidad y manteniendo pulsada la tecla '**abajo**' hasta llegar a la última entidad:



En World Editor Pro debes entrar en las propiedades de las entidades una a una.

6.- Ya sólo queda compilar el nivel y probarlo:



En esta ventana, no te olvides de cambiar el nombre del fichero .map que va a generar la compilación, en nuestro caso ponemos '**maz1.map**'.

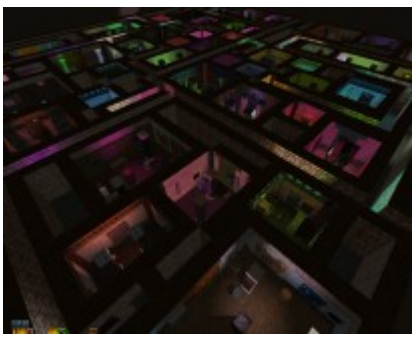
En este punto es conveniente volver a salvar el nivel, esta vez puedes usar el botón de salvar .

Generando laberintos:

Si en lugar de generar una mazmorra normal quieres generar una mazmorra laberíntica (sin habitaciones), no es necesario que uses el programa 'Dungeon Maker', simplemente empieza el proceso por la sección **(B)**, y en la sección **(C)** en lugar de usar el botón '**Generar mazmorra 'Dungeon Maker'**, usa el botón '**Generar mazmorra laberinto**'.
Las dimensiones del laberinto que deseas generar deberás establecerlas previamente en el fichero '**mazmorra.ini**' con estos dos parámetros:
ancho_laberinto = 31
largo_laberinto = 41
... en este caso para generar un laberinto de 31 x 41

Edición posterior de las mazmorras en World Editor:

Como es lógico, una vez tenemos la mazmorra en formato .3DT podemos cargarla con World Editor siempre que queramos para poder eliminar, modificar o añadir lo que nosotros queramos para adaptarla a nuestras necesidades y gustos.
Algunas de las cosas que podrías realizar en estas mazmorras podrían ser:
- Establecer propiedades de bloque 'window' a ciertos elementos 'brush' de los techos para que no produzcan sombras poco creibles.
- Eliminar bloques que impiden el paso del jugador a zonas vitales del mapa.
- Mover, añadir o eliminar bloques para que la mazamorra no presente un aspecto tan cúbico.
- Añadir enemigos, efectos de fuego, humo, etc... para dotar de más vida a la mazmorra.
- Crear 'areas' para aumentar el rendimiento general del nivel.
- Crear puertas en algunas habitaciones.
- Modificar las texturas del fichero '**mazmorra.txl**' que se encuentra en la carpeta '**levels**' para sustituir las texturas que ya existen por las que queramos nosotros. Es importante que no elimines ninguna textura, sólo sustitúyela por otra diferente, con el mismo nombre y del mismo tamaño.



Ficheros .LOG y .INI:
Fichero de informe de procesos de Entidad 3D

Informe_Entidad_3D.Log

```
Fichero de informe de procesos de Entidad 3D
--------------------------------------------

Versión de Entidad 3D: 3.2
Hora: 16:54:49

Inicio de procesos ...
Procesando parámetros de la línea de comandos
Parámetros OK
Creando procesos "Game Manager" y "Engine"
Procesos "Game Manager" y "Engine" OK
Creando lista de "drivers" de pantalla disponibles
Lista de "drivers" de pantalla disponibles OK
Seleccionando "driver" de pantalla
Selección "driver" de pantalla OK
Activando "driver" de pantalla
Activación "driver" de pantalla OK
Creando sistema de textos en pantalla
Creación sistema de textos en pantalla OK
Creando sistema de menú en pantalla
Creación sistema de menú en pantalla OK
Creando sistema de sonido del motor 3D
Creación sistema de sonido del motor 3D OK
Creando sistema "host" del motor 3D
Creación sistema "host" del motor 3D OK
Creando sistema de marcadores
Creación sistema de marcadores OK
Inicializando sistema de sonido BASS
Inicialización sistema de sonido BASS OK
Comprobando "drivers" Direct X
Comprobación "drivers" Direct X OK
Iniciando presentación "intro"
Presentación "intro" OK
Entrando en el bucle de nivel
```

Cada vez que el programa Entidad 3D ('entidad3d.exe') se pone en funcionamiento, crea en la carpeta principal de Entidad 3D un fichero llamado **'Informe_Entidad_3D.Log'** que utiliza el programa para ir almacenando los nombres de los diversos procesos internos que va realizando.
Este fichero .LOG de procesos lo podrás abrir con cualquier editor de textos y poder consultarlo en caso de que el programa falle sin presentar ningún error en pantalla, al menos podrás saber que es lo que el programa estaba haciendo o se disponía a hacer antes de producirse el error.
En algún caso te podrá servir de ayuda para poder saber donde está el error y poder modificar algo en las entidades o ficheros que intervengan en tu juego.
En caso de que tu PC presente algún tipo de incompatibilidad con Entidad 3D, podrás saber en que punto dejó de funcionar el programa e intentar solucionarlo tú mismo si es un problema de 'drivers' de pantalla o de tarjeta de sonido, por ejemplo.
Este fichero de procesos también puede resultar útil para el desarrollador del programa Entidad 3D para saber en que procesos se 'atasca' el programa y poder solucionar el problema en futuras actualizaciones de Entidad 3D.
Si te aparece algún error de este tipo en tu PC y realizas una consulta en el foro de Entidad 3D o lo comunicas al propio creador de Entidad 3D, es muy recomendable que adjuntes el fichero **'Informe_Entidad_3D.Log'** para poder investigar el error.
También sirve, junto con otros ficheros .LOG de otros usuarios, para llevar estadísticas de que puntos son los más críticos en el programa para que el creador de Entidad 3D pueda valorar las situaciones de error y decidir o no hacer algo en las próximas actualizaciones del programa, ya que hay que tener en cuenta que hay ocasiones en las que el programador de Entidad 3D podría solucionar el problema para las máquinas en las que aparecía el error, pero con ello, involuntariamente, hacer que aparezcan nuevos errores en máquinas que no aparecía ningún error.

Programador de Entidad 3D: Jordi Pérez
Foro de Entidad 3D: http://www.entidad-3d.com/foro/
Web de Entidad 3D: http://www.entidad-3d.com/

» *Consulta la sección 'Entidad3d.INI' para saber como hacer para que ciertos datos no se salven en el fichero '**Informe_Entidad_3D.Log**'.*
» *Sólo comentar que todas las versiones de Entidad 3D que se distribuyen han sido probadas en diversas máquinas, no sólo en la que se desarrolla el programa, y siempre teniendo en cuenta que no aparezca ningún error del tipo que se comenta en esta sección, aunque eso no quiere decir que te encuentres con alguno de estos errores, por las características particulares de la configuración de tu PC o simplemente porque el programa Entidad 3D no se probó suficientemente antes de distribuirlo.*

Fichero 'Entidad3d.ini'

El fichero de configuración '**entidad3d.ini**' contiene una serie de variables y valores que usa el programa Entidad 3D de manera interna para diversas funciones que realiza el programa antes de proceder con la carga del nivel.
Estos valores podemos modificarlos editando el fichero '**entidad3d.ini**' con cualquier editor de textos tipo Notepad o similar.
Los nombres de las variables que puede contener este fichero, sus valores y significados, se describen a continuación:
Leer_sistema
Con valor **1**, Entidad 3D realizará un chequeo al iniciar el programa para obtener información del PC que almacenará posteriormente en el fichero 'informe_entidad_3d.log'.
Con valor **0**, Entidad 3D no realizará este chequeo. Esto puede ser útil en PCs donde el programa Entidad 3D no arranque debido al acceso a datos internos del PC. Este problema no es habitual, pero de esta forma hay una posibilidad de saltarse este chequeo en caso de que impida iniciar correctamente Entidad 3D.
Valor recomendado: **1**
Retardo_intro
Permite definir, en segundos, el intervalo de tiempo entre cada una de las imágenes que se presentan en la introducción inicial del juego.
Puede ser un valor entre **0** y **59**. Si indicamos **0** no habrá intervalo de tiempo entre imágenes.
Valor recomendado: **5**
Liberar_memoria_basura
Poner a 1 si se desea que Entidad 3D libere memoria de Windows cada vez que se pase al siguiente nivel del juego.
Poner a 0 si se desea que Entidad 3D no libere memoria de Windows cada vez que se pase al siguiente nivel del juego.
Si observas en tu juego que al ir pasando de niveles cada vez te queda menos memoria libre en tu sistema, es conveniente usar esta opción (poniendo **Liberar_memoria_basura = 1**) para liberar memoria residual de Windows.
Valor recomendado: **0**

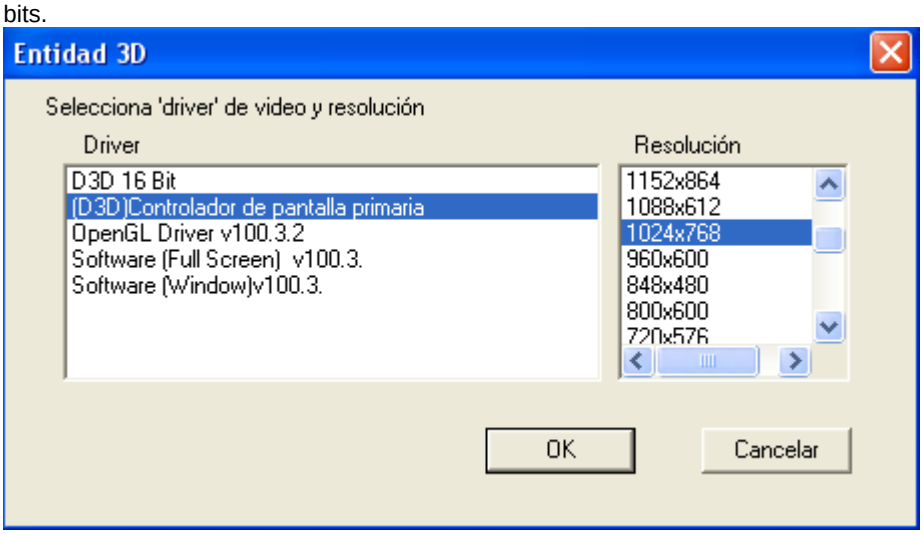
Si queremos definir un fichero de configuración .INI para un juego en concreto, sólo tenemos que crear el nombre del fichero comenzando por el nombre del primer nivel del juego:
*PrimerNivel*_entidad3d.ini
... para un nivel llamado por ejemplo **selva** el nombre del fichero de configuración sería:
selva_entidad3d.ini

» *Por motivos de seguridad, la variable '**Leer_sistema**' sólo será leída del fichero 'entidad3d.ini' y no de los posibles ficheros .ini que hayas creado para los niveles.*

Calidad del color

En este apartado veremos la manera de ejecutar nuestro juego con distinta calidad del color, lo que también se llama profundidad de color.

A partir de la versión 3.0 de Entidad 3D podemos jugar a nuestros juegos, aparte de a más resolución, con más calidad en los colores de pantalla.
Hasta ahora la profundidad de color que se utilizaba era de 16 bits, una calidad de color suficiente para la mayoría de personas, pero no para los más exigentes.
La posibilidad, también a partir de la versión 3.0, de utilizar texturas de 24 bits en nuestros escenarios, hacía del todo necesario poder utilizar los modos de pantalla que utilizaran esa calidad de color, por lo que a partir de la versión 3.0 es posible activar los modos de 16bits, 24bits e incluso de 32bits de color para las resoluciones de pantalla que seleccionemos.
Una cosa a tener muy en cuenta es que no todas las tarjetas gráficas soportan ni todas las profundidades de color ni todas las resoluciones, por lo que si intentas activar algún modo que tu tarjeta no soporta, el mismo programa te avisará.
Por defecto, incluso a partir de la versión 3.0 de Entidad 3D, los juegos se comenzarán a visualizar con una profundidad de color de 16bits y una resolución de 640x480. Esto es básicamente para ser compatible con la práctica totalidad de tarjetas gráficas y que por lo menos, de entrada, podamos ver nuestros primeros niveles, los niveles de ejemplo y los juegos de ejemplo sin ningún tipo de problemas.
Si te has leído la sección 'Preparando un acceso directo' ya sabrás como seleccionar el 'driver' y la resolución que utilizará Entidad 3D para un escenario en concreto, pero habrás visto que sigue sin haber una manera de seleccionar la profundidad de color. Pues la manera es editando un fichero que encontrarás en la carpeta principal de Entidad 3D (en la misma carpeta donde se encuentra el programa 'entidad3d.exe'). El fichero que tienes que editar se llama **D3D24.ini** y lo que debes modificar es el valor que está a la derecha de 'Bpp = '.
Por ejemplo: Si queremos que la próxima vez que lancemos Entidad 3D se vea con una profundidad de color de 24 bits, la línea Bpp quedaría de esta forma:
Bpp = 24
El resto de líneas no es necesario modificarlas. Salvas el fichero **D3D24.ini** y la próxima vez que lances Entidad 3D y selecciones el driver '**(D3D)Controlador de pantalla primaria**', tu escenario se presentará con la profundidad de color de 24 bits.



Pues ya sabes que debes poner para conseguir 32 bits de color:
Bpp = 32
Es importante que selecciones el driver de pantalla '**(D3D)Controlador de pantalla primaria**' para los modos de 24bits y 32bits, de lo contrario Entidad 3D utilizará el driver que usa por defecto 'D3D 16 Bit', que trabaja a 16 bits de color aunque hayamos modificado el Bpp del fichero 'D3D24.ini'.

» Si utilizas el driver de pantalla OpenGL ('OpenGL Driver' en la selección de drivers), debes tener en cuenta que de todos los datos que puedes modificar en el fichero D3D24.INI, sólo se usarán Bpp y ZBufferD, el resto sólo son utilizados por los drivers de Direct 3D ('D3D' en la selección de drivers).
» Si distribuyes tu juego, asegúrate de que la distribución contenga el fichero D3D24.INI con sus valores originales, sobre todo el Bpp = 16 ya que los modos 24 y 32 no son 100% compatibles con todas las tarjetas gráficas.

Mazmorra.ini

(Configuración del generador de mazmorras y laberintos)

El juego de ejemplo 'Secta'

Modificación del juego

Juego 'Secta'

'Secta' es el nombre del juego de ejemplo que acompaña a Entidad 3D v2.0 y al que podrás jugar desde el primer momento con sólo ejecutar el fichero 'secta.bat'.
Eres libre de alterar este juego para modificar lo que quieras, pero piensa que borrar o modificar ciertos ficheros de las carpetas 'levels', 'actors', 'objects', 'bmp', 'comandos' o 'wav', podría hacer que el juego dejase de funcionar.
Si queremos asegurarnos de conservar una copia intacta del juego, pues no nos queda más remedio que realizar una copia de seguridad de todo el directorio donde hemos instalado Entidad 3D (por defecto el directorio 'c:\entidad_3d') en algún otro directorio de tu disco duro. De esta forma tendrás la carpeta original para alterar lo que quieras y si quieres jugar a 'Secta' te vas a tu copia de seguridad y ejecutas allí 'secta.bat'.
Eso si, si en este punto quieres editar o modificar el juego 'Secta' que tienes en la copia de seguridad, tendrás que utilizar todo lo que hay en esa copia, incluido el editor de niveles 'World Editor', ejecutando el programa 'gedit.exe' que hay en esa copia de seguridad o 'tpack.exe' para el caso del programa 'Texture Packer'.
Puede parecerte un lío, pero es la única manera de asegurarte que no alteras en absoluto el juego de ejemplo que siempre te podrá servir de guía o de base inicial en la creación de tu juego que estarás realizando en la carpeta original.

El único nivel que compone el juego 'Secta' es el que podrás encontrar en la carpeta 'levels' con el nombre 'secta.3dt' (que es el fichero que podrás cargar con 'World Editor') y 'secta.bsp' que es el nivel ya compilado con 'World Editor' y que es el que utiliza directamente Entidad 3D.
Este último fichero he de decirte que está compilado con la opción 'Full Vis' activada, cosa que no se hace normalmente cuando se está desarrollando un nivel ya que este tipo de compilación si bien deja un .BSP muy optimizado para obtener buenos rendimientos en tiempo de juego, es una compilación muy muy lenta. Sólo decirte que un nivel como el de 'Secta' compilado de esta forma tardó 10 horas seguidas en un Pentium IV a 3.2Ghz

Controles del juego

Si bien dentro del juego el jugador puede configurar los controles del juego, la configuración por defecto es la mayoritariamente utilizada en este tipo de juegos 3D.
Se trata de una combinación de teclado y ratón para agilizar todos los movimientos del personaje protagonista.

Controles por defecto en 'Entidad 3D'. Esta configuración se guarda en el fichero '**teclas.dat**':
Disparar: Botón izquierdo del ratón
Saltar: Botón derecho del ratón
Hacia la izquierda: Cursor izquierda
Hacia la derecha: Cursor derecha
Hacia adelante: Cursor arriba
Hacia atrás: Cursor abajo
Agacharse: Tecla 'C'
Ponerse de pie: Tecla 'V'
Activa/Desactiva el modo volar: Tecla 'F'
Selección siguiente arma: Tecla 'Control'

Selección arma Blaster: Tecla '1'
Selección arma Lanza-granadas: Tecla '2'
Selección arma Lanza-cohetes: Tecla '3'
Selección arma Fusil: Tecla '4'
Mirar arriba: 'Tecla 8 de teclado numérico'
Mirar abajo: 'Tecla 2 de teclado numérico'
Girar derecha: 'Tecla 3 de teclado numérico'
Girar izquierda: 'Tecla 1 de teclado numérico'
Vista en primera persona: Tecla 'Q'
Vista en tercera persona: Tecla 'A'
Vista estilo RPG: Tecla 'Z'
Vista estilo Plataformas: Tecla 'X'
Menú del juego: Tecla 'ESC'
Muestra consola: Tecla 'Tabulador'
Panel de objetivos: Tecla 'Borrar hacia atrás'
Inventario de objetos y menú de acciones: Tecla 'i'
Selección de objetos en inventario: Teclas de flechas izquierda y derecha del teclado numérico
Selección de acción en el menú de acciones: Teclas de flechas arriba y abajo del teclado numérico
Captura de pantalla: Tecla 'F12' (las capturas se guardan en la carpeta 'capturas')
Pausa y reanuda juego: Tecla 'P'
Activa punto de mira: Tecla '+'
Desactiva punto de mira: Tecla '-'
Tecla 'Acción': Tecla 'ESPACIO'
Activa fuente de luz si el jugador la lleva: Tecla 'M'
Desactiva fuente de luz si el jugador la lleva: Tecla 'N'
Una vez muerto, para utilizar la siguiente vida: Pulsa 'Saltar'
Movimiento del ratón: 'Permite controlar la dirección hacia donde mira el jugador'
Invertir ratón: 'Permite invertir el movimiento que realiza el ratón'

» *Dentro del menú de juego existe la posibilidad de configurar todas las teclas.*

Uso del Gamepad o Joystick

A continuación se describen las funciones que soporta Entidad 3D si se utiliza un mando de juegos.

Cruceta o pad analógico izquierdo:
Movimiento del jugador
Pad analógico derecho:
Rotación de la vista del jugador
Botón 1:
Disparar
Botón 2:
Saltar
Botón 3:
Siguiente arma
Botón 4:
Acción
Botón 5:
Selección arma Blaster
Botón 6:
Selección arma Lanza-granadas
Botón 7:
Selección arma Lanza-cohetes
Botón 8:
Selección arma Fusil

Funcionamiento del juego fuera de 'World Editor':

Preparando un acceso directo

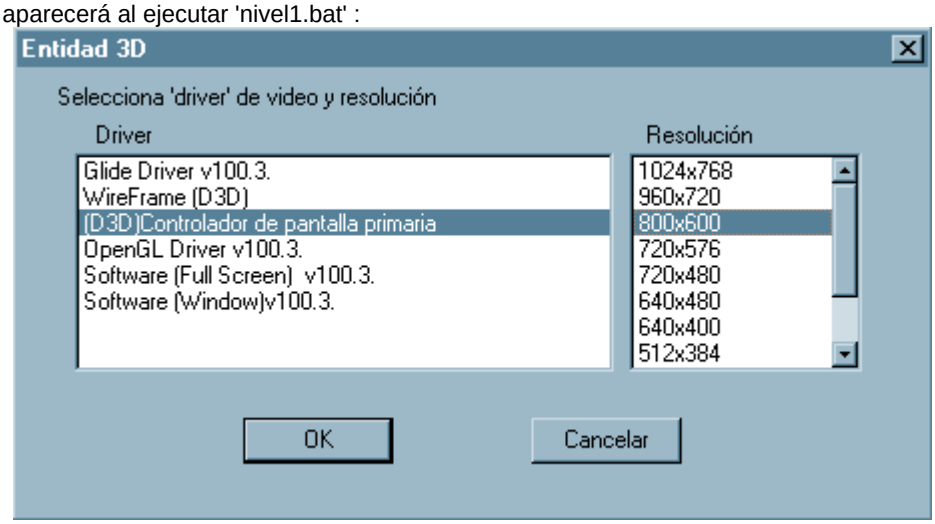
En esta sección veremos la manera de ejecutar nuestro juego desde fuera de 'World Editor'.

Hasta ahora hemos cargado nuestro escenario de pruebas desde dentro del propio programa 'World Editor'. Justo después de compilarlo se ejecutaba para poder probarlo tal y como lo haría el jugador.
Esto está muy bien para poder ver y probar nuestro escenario durante el proceso de desarrollo del juego, pero evidentemente necesitamos una manera de ejecutar nuestro juego de manera instantánea para que el usuario del juego lo ponga en marcha de una manera fácil y rápida, a parte de que nuestra distribución del juego no contendrá el 'World Editor', ya que el jugador no lo necesitará para nada.
Para hacer que nuestro juego se ejecute con un solo doble 'click' de ratón tendremos que preparar un archivo .BAT en la carpeta principal de 'Entidad 3D' de la siguiente forma:
Supongamos que nuestro escenario creado con 'World Editor' se llama 'nivel1'. Con cualquier editor de textos, crearemos un fichero que se llamará 'nivel1.bat' y dentro de este pondremos las siguientes lineas:
Entidad3d.exe -name Jugador -map nivel1
exit
'-name Jugador' no es necesario que lo cambies nunca, mientras que detrás de '-map' tendrás que poner el nombre del escenario a cargar.
En la carpeta principal de 'Entidad 3D' (por defecto en 'c:\entidad_3d') nos aparecerá el siguiente icono:

nivel1.bat

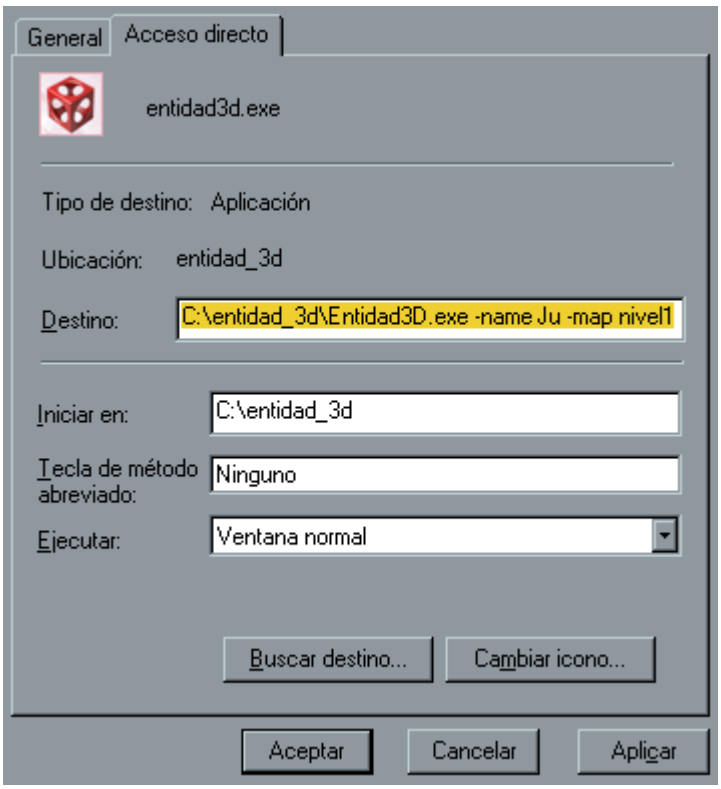
Este acceso es el que nos servirá para dar comienzo nuestro juego de forma independiente de 'World Editor'.
Podemos crear distintos ficheros .BAT para lanzar diferentes escenarios de forma independiente.

Si queremos que antes de dar comienzo el juego nos aparezca una ventana para poder seleccionar el 'driver' y resolución de pantalla, el fichero 'nivel.bat' quedaría así:
Entidad3d.exe -name Jugador -map nivel1 -PickMode
exit
Esto es lo que nos aparecerá al ejecutar 'nivel1.bat' :



Si no utilizamos el parámetro '-PickMode' nuestro juego comenzará a funcionar por defecto con los drivers 'Microsoft Direct 3D' a una resolución de 640x480'.
De todas formas, una vez en el juego podremos cambiar el 'driver' y resolución en uso a través del menú que aparece al pulsar la tecla 'ESC'.
Si queremos dar comienzo el juego sin que pregunte la resolución y Entidad 3D use el 'driver' de pantalla y resolución que el jugador seleccionó la última vez, usaremos lo siguiente:
Entidad3d.exe -name Jugador -map nivel1 -PrevMode
exit
Estos ficheros '.bat' nos facilitan la tarea de proporcionar al jugador una manera rápida y fácil de acceder a nuestro juego, pero es posible que no te acabe de convencer eso de hacer un '.bat' de MSDOS para correr un juego de Windows. En realidad este salto momentáneo a MSDOS no tiene que implicar ningún problema y en realidad el juego al final correrá completamente en Windows.
Pero si no quieres lanzar tus niveles a base de ficheros '.bat', siempre podrás crear accesos directos de Windows para lanzar los niveles, de esta forma el juego no abrirá esa sesión de MSDOS.
Así es como quedarían los datos de un acceso directo en Windows para, por ejemplo, lanzar el 'nivel1':
Pero si deseas no utilizar ningún fichero .BAT, que por otro lado usar este sistema que utiliza el MS-DOS no es muy elegante si se supone que estamos realizando un juego para Windows, siempre te queda la posibilidad de crear un fichero que tendrás

que llamar **autonivel.ini** y que tendrá que estar en la carpeta donde tenemos los niveles, es decir la carpeta **levels** de Entidad 3D:



¿ Y que debe contener ese fichero ?, pues exactamente lo que en los accesos directos ponemos detrás de 'entidad3d.exe'. Para el ejemplo anterior crearíamos, con cualquier editor de textos, un fichero llamado **autonivel.ini** que contendría lo siguiente:
-name Jugador -map nivel1 -PickMode
De esta forma, cuando el jugador ejecute directamente el programa **entidad3d.exe** se recogerán como parámetros lo que hayamos puesto en el fichero **autonivel.ini**, en este caso al haber puesto '-PickMode' primero se solicitará al jugador el 'driver' y resolución de pantalla y a continuación se lanzará el nivel llamado 'nivel1'.
Si se ejecuta **entidad3d.exe** y no tenemos ningún fichero 'autonivel.ini' creado, Entidad 3D intentará lanzar el juego de ejemplo 'Secta'.
Si lo deseas puedes cambiar el nombre de **entidad3d.exe** por alguno que vaya más acorde con tu juego, por ejemplo 'legend.exe', el programa funcionará exactamente igual pero conseguirás dejar completamente personalizado tu juego.
Pero ten en cuenta que si cambias el nombre del .EXE de Entidad 3D, el editor de niveles World Editor no podrá lanzar los niveles ya que este programa busca 'entidad3d.exe' para funcionar. Si quieres solucionar esto tienes que editar el fichero 'gedit.ini' que hay en la carpeta principal de Entidad 3D y donde veas 'entidad3d.exe' sustitúyelo por en nombre que le hayas dado al .EXE

Empaquetado del juego en un archivo .ZIP

En esta sección veremos como reducir al máximo el número de ficheros que utiliza nuestro juego para poder hacer un archivo .ZIP independiente de todo el entorno de desarrollo del juego para que podamos distribuirlo como un juego completo.

En primer lugar lo que haremos será ejecutar el programa 'distribucion.bat' que encontrarás en la carpeta principal de 'Entidad 3D'. Este fichero .BAT te creará una carpeta llamada 'distribucion' que contendrá una copia de todos los ficheros necesarios para el correcto funcionamiento de nuestros juegos fuera de nuestro ordenador.

distribucion.bat

Efectivamente están todos los ficheros necesarios, pero también habrá algunos que no serán necesarios y que sólo tu sabrás los que son y que deberás eliminar:
¿ Qué ficheros son los que debes borrar ?:
En la carpeta 'distribucion/': Los ficheros .BAT que no pertenezcan al juego que deseamos distribuir.
En la carpeta 'distribucion/Actors': Los ficheros .ACT que no pertenezcan al juego que deseamos distribuir.
En la carpeta 'distribucion/Objects': Los ficheros .ACT que no pertenezcan al juego que deseamos distribuir.
En la carpeta 'distribucion/Bmp': Los ficheros .BMP que no pertenezcan al juego que deseamos distribuir.
En la carpeta 'distribucion/Wav': Los ficheros de audio que no pertenezcan al juego que deseamos distribuir.
En la carpeta 'distribucion/Comandos': Los ficheros .INI o .DLG que no pertenezcan al juego que deseamos distribuir.
En la carpeta 'distribucion/Levels': Los ficheros .BSP que no pertenezcan a escenarios del juego que deseamos distribuir.
En la carpeta 'distribucion/Levels': Todos los ficheros .TXL de texturas. Si ... todos, para jugar al juego no se necesita ninguno, ni siquiera el de tu juego. Tu nivel junto con el fichero de texturas ya están incluidos en el interior del fichero .BSP .
ATENCION:
.En todas estas carpetas no borres los ficheros que aunque no son necesariamente de tu juego, Entidad 3D los necesita, como son los modelos .ACT de las armas, salud, munición, etc.., gráficos .BMP o sonidos .WAV, así como el objeto número uno '1_caja.act'.

Después de esta limpieza sólo tendremos en la carpeta 'distribución' los ficheros que necesita nuestro juego para funcionar de forma independiente.
Sólo nos queda empaquetar todo el contenido de esta carpeta (con todas las carpetas que cuelgan de esta) en un fichero .ZIP para poder distribuirlo por Internet o simplemente para copiarlo en otro ordenador.
Para ejecutar el juego sólo habrá que descomprimir todo el archivo .ZIP en un disco duro y utilizar el .BAT que hemos creado para dar comienzo el juego.

... pues hasta aquí ha llegado este tutorial/manual de 'Entidad 3D', de momento, y recuerda.... *'los buenos juegos no dependen necesariamente de la tecnología disponible, dependen de la creatividad y de la imaginación'.*
Suerte.
// (C)Febrero 2005 Jordi Pérez \\ eMail: jordi_sbd@terra.es

Programas que incluye la última distribución de Entidad 3D:

World Editor 1.0

World Editor 2.0

World Editor Pro

Texture Packer

Actor Viewer

Entidad 3D Actor Viewer

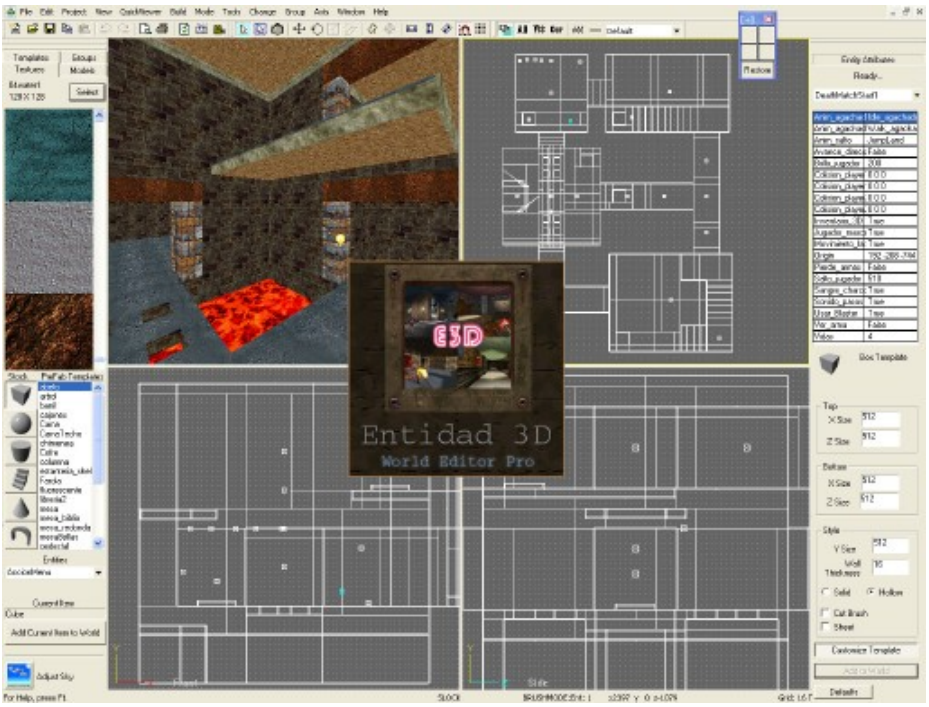
Actor Studio

Skin replacer

Dungeon Maker

Entidad 3D

World Editor Pro v0.7



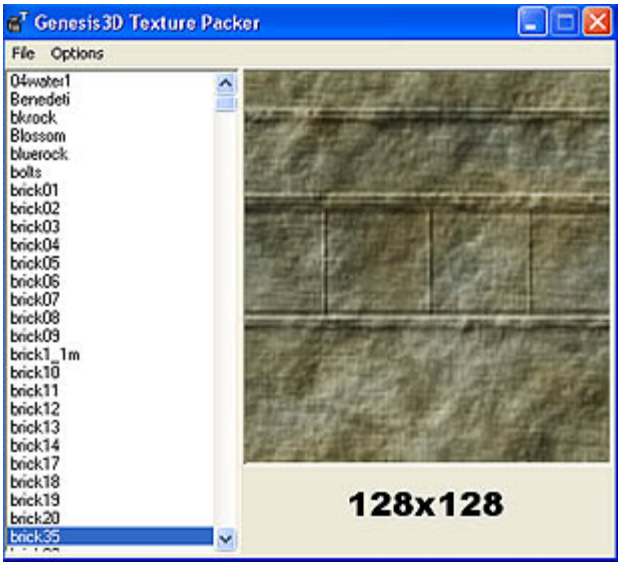
¿ Donde se encuentra ?:
C:\ENTIDAD_3D\WorldEditorPro.exe
¿ Para que sirve ?:
Este editor sirve exactamente para lo mismo que World Editor 1.0 y World Editor 2.0, para crear los escenarios, pero esta versión es la más actualizada y presenta diversas mejoras:

- Interface de usuario muy mejorado. Más elementos en pantalla sin tener que acceder a la barra de menú.
- Los modelos .ACT de la entidad 'SpecificActor' se ven en cualquiera de las vistas.
- Dispone de opción deshacer (Undo).
- Mediante un sólo 'click' nos permite trabajar con una de las cuatro vistas a máximo tamaño.
- La rueda del ratón permite realizar zooms en cualquiera de las cuatro vistas.

¡¡ ATENCION !! Si bien los ficheros .3DT de niveles creados con World Editor 1.0 o World Editor 2.0 se pueden cargar perfectamente con World Editor Pro, los ficheros .3DT de niveles creados con World Editor Pro **no se pueden cargar** en World Editor 1.0 ni en World Editor 2.0.
¿ Como funciona ?:
Consulta esta guía rápida para familiarizarte con este editor.
Se recomienda comenzar a usar este editor sólo a gente que ya haya usado la versión 1.0 o 2.0 de World Editor.

World Editor Pro v0.7 es un programa de código abierto desarrollado por WildTangent, Inc.

tpack Texture Packer



¿ Donde se encuentra ?:
C:\ENTIDAD_3D\tpack.exe
¿ Para que sirve ?:
Se utiliza para añadir o eliminar texturas de las librerías de texturas que utiliza World Editor. Estas librerías son ficheros .txl que podrás encontrar en la carpeta 'levels' de Entidad 3D.
¿ Como funciona ?:
Podrás encontrar información del funcionamiento de este programa en la sección 'Añadiendo nuevas texturas' del tutorial de World Editor incluido con Entidad 3D, este tutorial está en formato HTML, utiliza tu navegador de Internet para visualizar este tutorial. Por defecto tienes que tenerlo en:
C:\ENTIDAD_3D\tutorial_world_editor\index.html.
También puedes consultar el tutorial llamado 'Tutorial para crear fondos de exteriores para Entidad 3D' que se encuentra en la web de Entidad 3D, donde se explica como crear fondos para tu juego utilizando 6 texturas añadidas con Texture Packer a tu librería de texturas.

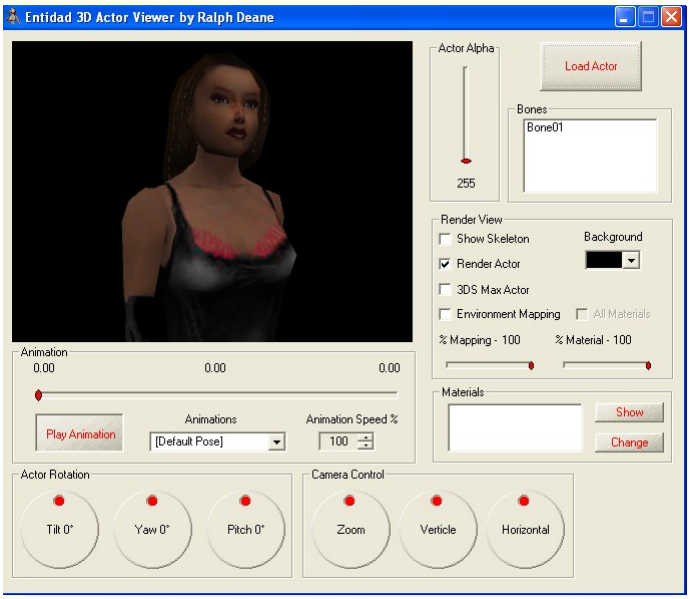
'Texture Packer' es propiedad de Eclipse Entertainment y se distribuye bajo licencia 'Open Source', junto con todo su código fuente, con Genesis3D. Para más información visitar la página de Genesis3D.

ActView Actor Viewer

¿ Donde se encuentra ?:
C:\ENTIDAD_3D\ActView.exe
¿ Para que sirve ?:
Es una utilidad que nos permite visualizar los modelos 3D en formato .act que tenemos tanto en las carpetas 'actors', 'objects' y 'armas' de Entidad 3D.
Sólo sirve para poder ver los objetos o personajes, no para modificarlos.
¿ Como funciona ?: Sólo tienes que ejecutar esta utilidad, seleccionar el 'driver' de pantalla a utilizar (preferiblemente D3D) y abrir cualquier fichero .act con la opción 'File/Open'. Una vez cargado el modelo dispones de diversos controles para poder girarlo, hacerle zoom, moverlo o ver sus animaciones.

'Actor Viewer' es propiedad de Eclipse Entertainment y se distribuye bajo licencia 'Open Source', junto con todo su código fuente, con Genesis3D. Para más información visitar la página de Genesis3D.

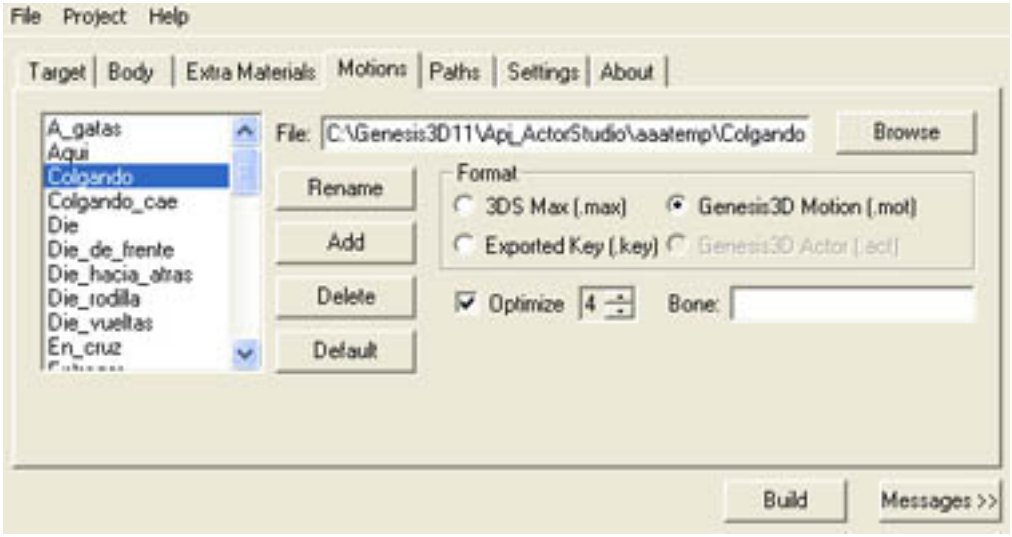
Entidad 3D Actor Viewer



¿ Donde se encuentra ?:
C:\ENTIDAD_3D\Viewer\Viewer.exe
¿ Para que sirve ?:
Al igual que el 'Actor Viewer', es una utilidad que nos permite visualizar los modelos 3D en formato .act que tenemos en las carpetas 'actors', 'objects' y 'armas' de Entidad 3D.
Este programa también permite reasignar las texturas que componen el modelo 3D.
¿ Como funciona ?: Sólo tienes que ejecutar esta utilidad y pulsar el botón 'Load Actor' para abrir cualquier fichero .act. Una vez cargado el modelo dispones de diversos controles para poder girarlo, hacerle zoom, moverlo o ver sus animaciones, entre otras cosas.

'Entidad 3D Actor Viewer' es un programa de código abierto desarrollado por Ralph Deane.

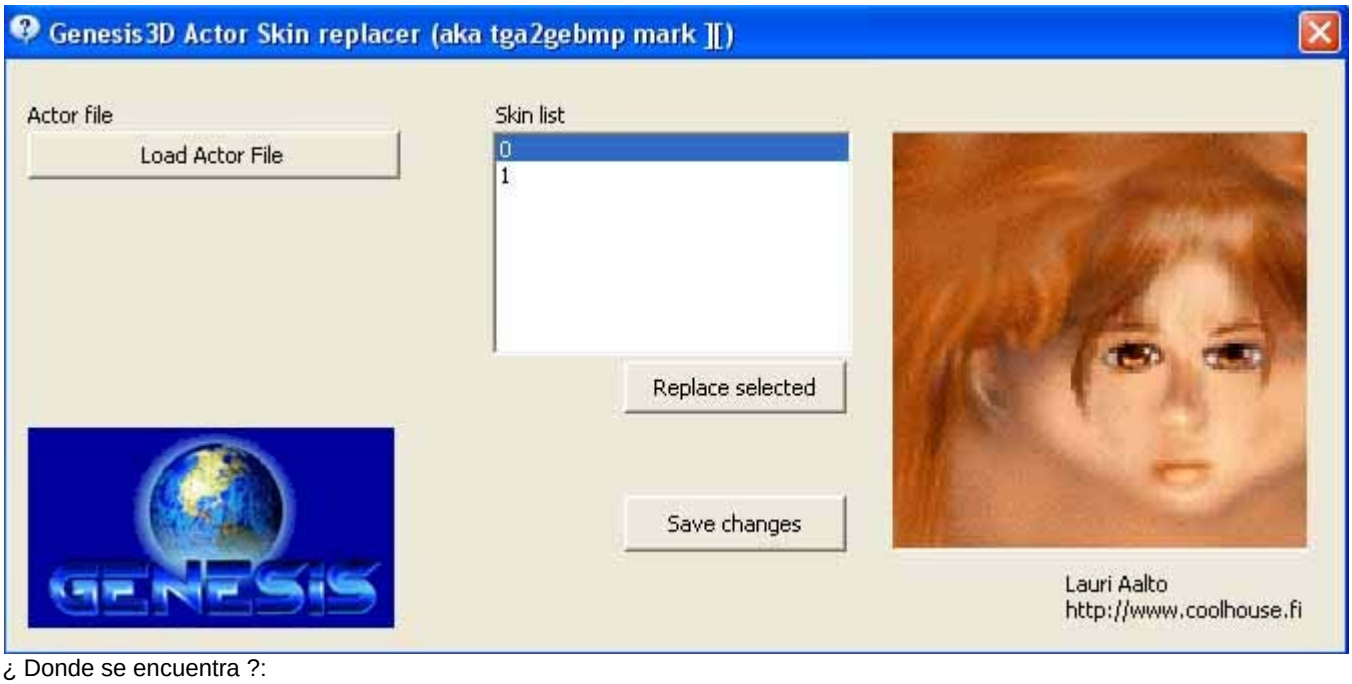
AStudio Actor Studio



¿ Donde se encuentra ?:
C:\ENTIDAD_3D\AStudio.exe
¿ Para que sirve ?:
Pues para crear modelos 3D en formato .act (que es el utilizado por Entidad 3D) a partir de ficheros de polígonos .BDY (creados con el programa MilkShape 3D) y ficheros de animaciones .MOT.
¿ Como funciona ?:
Consulta el tutorial llamado 'Tutorial para convertir personajes de Half-Life a personajes de Entidad 3D' que podrás encontrar en la página de Entidad 3D. Aquí se hace un uso práctico de Actor Studio para convertir personajes en formato del juego Half Life al formato .act.

'Actor Studio' es propiedad de Eclipse Entertainment y se distribuye bajo licencia 'Open Source', junto con todo su código fuente, con Genesis3D. Para más información visitar la página de Genesis3D.

Skin replacer

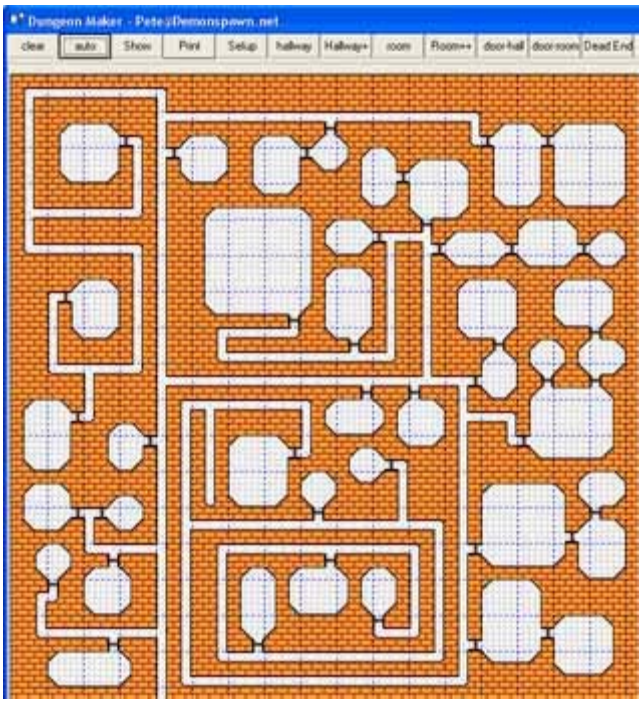


¿ Donde se encuentra ?:
C:\ENTIDAD_3D\skin_replacer\tga2gebmp2.exe
¿ Para que sirve ?:
Nos permite cambiar las texturas de los modelos .ACT por otras en formato .BMP que tengamos en el disco duro.
¿ Como funciona ?:
- Ejecuta esta utilidad y pulsa el botón 'Load Actor File' para abrir el fichero .ACT que deseas modificar.
- Selecciona la textura que deseas cambiar en la caja 'Skin list'.
- Pulsa el botón 'Replace selected' y selecciona el fichero .BMP con la nueva textura.
- Finalmente pulsa el botón 'Save changes' para salvar los cambios realizados.

(Recuerda que las dimensiones de ancho y alto de la nueva textura debe ser alguna de las siguientes: 16, 32, 64, 128 o 256)

'Genesis3D Actor Skin replacer' es un programa de código abierto desarrollado por Lauri Aalto.

Dungeon Maker



¿ Donde se encuentra ?:
C:\ENTIDAD_3D\DungeonMaker\Dungenmk.exe
¿ Para que sirve ?:
Nos permite generar sencillas mazmorras al azar en 2D, que posteriormente Entidad 3D se encargará de generar en 3D.
¿ Como funciona ?:
Su funcionamiento se describe en el anexo 'Generación de mazmorras' de este mismo manual.

'Dungeon Maker' es un programa gratuito desarrollado por Peter Tyjewski.

Entidad 3D

¿ Donde se encuentra ?:
C:\ENTIDAD_3D\Entidad3D.exe
¿ Para que sirve ?:
Sirve para correr los niveles que componen nuestro juego, estos niveles los habremos desarrollado y compilado con World Editor para crear un fichero .BSP por nivel, son estos ficheros .BSP los que procesa Entidad 3D para finalmente poder jugar a los juegos en pantalla completa.
¿ Como funciona ?:
Este programa no lo ejecutes tal cual, a menos que uses el fichero autonivel.ini, porque no te funcionará, es necesario utilizar ficheros .bat o crear accesos directos donde se indique el nivel a cargar entre otros parámetros. Consulta la sección 'Preparando un acceso directo' de este mismo manual para saber como lanzar adecuadamente los niveles.
'Entidad3d.exe' también lo llama World Editor cuando utilizamos la opción de 'Preview' en la ventana de compilación de nuestro nivel, ya que una vez World Editor acaba de compilar el nivel, lo corre a pantalla completa llamando a 'Entidad3D.exe'.

Historial de versiones:

Entidad 3D v1.1
(Distribuido el 23/Marzo/2004)

En **Entidad 3D v2.0**
(Nuevas características con respecto a Entidad 3D v1.1)

En **Entidad 3D v2.1**
(Nuevas características con respecto a Entidad 3D v2.0)

En **Entidad 3D v2.2**
(Nuevas características con respecto a Entidad 3D v2.1)

En **Entidad 3D v2.3a**
(Nuevas características con respecto a Entidad 3D v2.2)

En **Entidad 3D v2.4**
(Nuevas características con respecto a Entidad 3D v2.3a)

En **Entidad 3D v3.0**
(Nuevas características con respecto a Entidad 3D v2.4)

En **Entidad 3D v3.2**
(Nuevas características con respecto a Entidad 3D v3.0)

En **Entidad 3D v3.3**
(Nuevas características con respecto a Entidad 3D v3.2)

En **Entidad 3D v3.4**
(Nuevas características con respecto a Entidad 3D v3.3)

En **Entidad 3D v3.4c**
(Nuevas características con respecto a Entidad 3D v3.4)

En **Entidad 3D v3.5a**
(Nuevas características con respecto a Entidad 3D v3.4c)

En **Entidad 3D v3.6a**
(Nuevas características con respecto a Entidad 3D v3.5a)

En **Entidad 3D v3.6b**
(Nuevas características con respecto a Entidad 3D v3.6a)

En **Entidad 3D v3.7a**
(Nuevas características con respecto a Entidad 3D v3.6b)

En **Entidad 3D v3.7d**
(Nuevas características con respecto a Entidad 3D v3.7a)

Nuevas características de Entidad 3D v2.0

A continuación se describen todas las nuevas características que se han implementado en la versión 2.0 con respecto a la versión 1.1:

- En la vista en primera persona, eliminación de los cambios de altura de la cámara al caminar el jugador.
- Corrección de orientación para armas y munición.
- En primera persona, visualización del arma correspondiente en frente del jugador. También incluye el efecto del fogonazo en la bocacha del fusil.

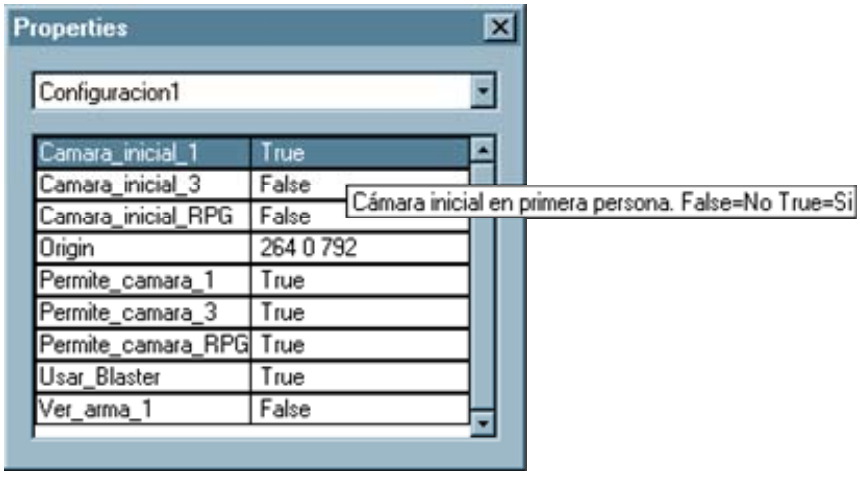
- Rotación del lanza-granadas cuando está en el suelo.
- Detener la música de fondo al pulsar la tecla de pausa.

- Una tercera cámara con una vista similar a los juegos de RPG. Esta cámara no provoca 'clipping' al colisionar con las paredes y sólo atraviesa una pared si ha de seguir al personaje del jugador.

- Nueva entidad llamada 'Configuracion' que permite el control de diversas características de nuestro juego:

Esta entidad entre otras cosas permite:

Establecer la cámara inicial en nuestro juego.
No permitir cambios de cámara o sólo permitir los cambios a las cámaras que designemos.
Permitir, o no, el uso del arma Blaster para aventuras sin armas.



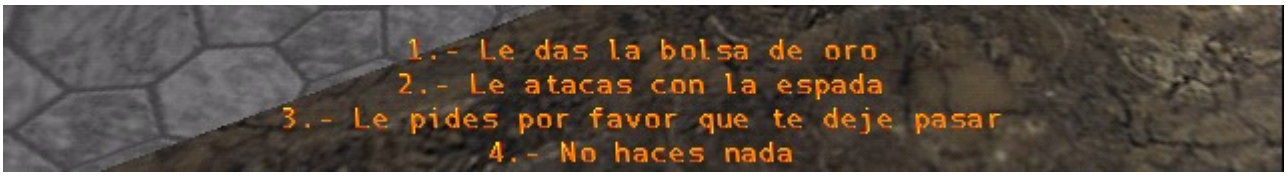
Establecer si se ve, o no, el arma del jugador.
Posibilidad de modificar el número inicial de vidas de que dispone el jugador.
Activar/Desactivar el sonido de pasos del jugador al correr.
Selección del género del jugador (masculino o femenino) para utilizar de forma automática sus sonidos correspondientes para salto, salud, daño y muerte.
Activar/Desactivar el uso del Joystick o Gamepad en el juego.
Establecer el volumen de la música de fondo.
Establecer si el jugador pierde sus armas al morir o no.

El jugador puede recoger un objeto que le proporcionará una fuente de luz que le acompañará. Siempre que disponga de esta luz podrá apagarla o encenderla a voluntad:

El arma del jugador también puede verse al utilizar cualquiera de las cámaras en tercera persona:

Cambios en los nombres de los ficheros de sonido que Entidad 3D utiliza para la introducción y para el fondo. En la versión 1.1 eran siempre los mismos nombres (intro.* y fondo.*). Esto tenía el inconveniente de que siempre se interpretaban las mismas músicas aunque nosotros cambiásemos de nivel, tanto en el nivel inicial como en niveles posteriores que cargásemos a partir de otro nivel.
Para evitar esto, ahora estos nombres de ficheros de audio tendrán que comenzar con el nombre del nivel.
Por ejemplo: Si tenemos el nivel llamado 'cueva.bsp', para que suene una música de introducción sólo para este nivel, prepararemos un fichero de audio que por ejemplo se llamaría 'cueva_intro.mp3' y para el fondo 'cueva_fondo.mod'.
Se ha solucionado de la misma forma para diversos ficheros que eran comunes y que ahora podrán diferenciarse en función del nivel que se cargue.

En cualquier momento se pueden plantear opciones al jugador. A cada opción podemos asociarle un fichero de comandos (.ini) distinto.
Entre múltiples utilidades, esta característica permite al jugador mantener auténticos diálogos con los personajes:



Posibilidad de poner enemigos estáticos. No se desplazan por el escenario pero si que disparan al jugador.
Este tipo de enemigo lo podemos utilizar para defender una puerta, una posición, un objeto, etc..
También puede ser útil para que haga de francotirador.
Otra posibilidad es que a un enemigo de estas características no le asociemos un personaje sino un arma, por ejemplo un cañón automático que proteja una zona.

Posibilidad de presentar una imagen BMP mientras se carga el nivel. Una imagen distinta para cada nivel:

Posibilidad de presentar una imagen BMP antes de cargar el nivel. Una imagen distinta para cada nivel. El jugador podrá ver esta imagen todo el tiempo que quiera hasta que pulse el botón izquierdo del ratón, la tecla de espacio o la tecla 'Enter'.
Esta imagen se puede utilizar para introducir al jugador en la temática de ese nivel, para indicar por que nivel va mostrando los niveles anteriores ya completados o para mostrar una portada artística que tenga que ver con el nivel:

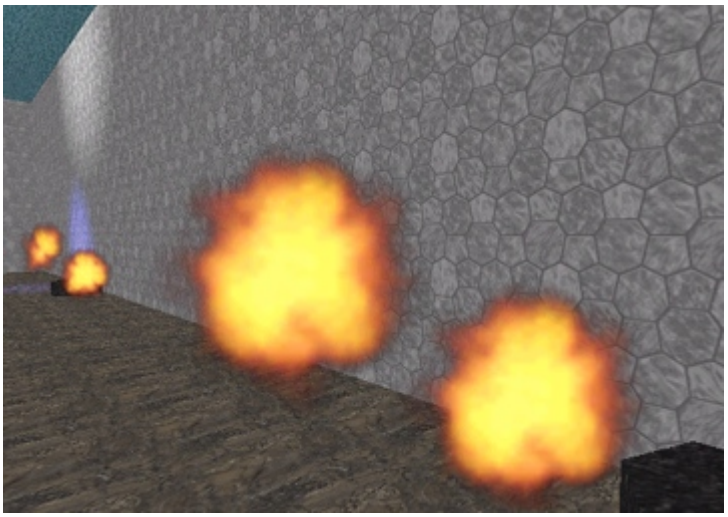
En cualquier momento se pueden superponer en pantalla imágenes 2D fijas.
Esto puede servir para mostrar fotos, planos, mapas, indicaciones, avisos, marcos, etc..
Estas imágenes soportan color transparente, se pueden poner varias simultáneamente, de cualquier tamaño, en cualquier sitio y podrán desaparecer cuando nosotros queramos o transcurrido un tiempo que podremos fijar:

Soporte para 'Gamepads' y 'Joysticks'.
Se aprovechan los 'pads' analógicos para una mejor precisión en la velocidad de giro de la cámara y en la velocidad de movimiento del personaje.
Se pueden utilizar hasta 8 botones que disponga el mando, aunque también se puede jugar con un simple 'pad' con 2 botones.

Nuevos comandos para manejar temporizadores para poder activar ficheros de comandos (.ini) después de transcurrido un tiempo.
Puede servir para hacer explotar bombas retardadas, presentar un mensaje al cabo de un tiempo o durante un tiempo, para permitir abrir una puerta en un determinado tiempo, etc... Todo dependerá de la combinación que realices con estos temporizadores y los comandos disponibles en Entidad 3D.

Se puede precisar mejor la zona de bloqueo que provoca cualquier 'actor', mediante un cubo azul transparente que podremos hacer aparecer mientras desarrollamos el juego y cuyas dimensiones podrán modificarse.

Nueva entidad llamada 'Explosion' que nos permite realizar explosiones animadas en determinados sitios y en determinados momentos.
Se puede modificar el tamaño de la explosión y su velocidad.
Tanto la animación de la explosión (ficheros .BMP con 'Alpha Blending') como el fichero de audio de la explosión (.WAV), puedes cambiarlos para crear tus propios efectos.



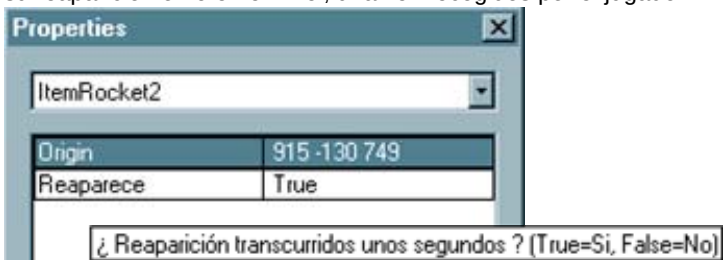
Se incluye el botón 'Acción'.
Podemos hacer que diversas acciones que tenga que realizar el jugador sólo funcionen si se encuentra en el lugar adecuado y al mismo tiempo pulsa la tecla 'Acción' (o un botón del 'gamepad').

Nuevo elemento 'ItemVida' para añadir a los escenarios. Este objeto proporcionará una vida extra al jugador.

Corregido el error que no permitía variar el volumen de la música de fondo cuando era un fichero de audio MP3.

Se ha elevado la altura de los escalones que sube automáticamente el jugador.
En la versión 1.1 si los escalones eran un poco altos el personaje no los subía y tenía que subir esas escaleras a base de saltos hacia adelante.

Los elementos (Salud, escudo, armas y municiones) puede configurarse su reaparición o no en el nivel, una vez recogidos por el jugador.



El arma 4 del enemigo (fusil ametrallador) produce ahora un fogonazo cada vez que se dispara, tanto del jugador como del enemigo. De esta forma por lo menos podemos ver quien nos está disparando, ya que las balas, al no verse venir, no se sabía de donde procedían.
La solución para que los enemigos dispongan de armas, pasa por tener que diseñar enemigos que ya lleven las armas 'enganchadas' en la mano derecha, por lo que hay que crear ficheros de personajes .act con enemigos que ya lleven un arma.

A continuación puedes ver 3 pantallas de ejemplo con este tipo de enemigos:

Se incluye la posibilidad de 'poner' sonidos 3D en el escenario.

Estos sonidos son posicionales 3D reales, escuchándose en el lado del altavoz que se encuentre el sonido y con la atenuación correspondiente según la distancia del jugador.
También se incluyen nuevos comandos para activar y desactivar estos sonidos durante el desarrollo del juego.

Se amplían las posibilidades de los comandos 'VIS0' y 'VIS1' para poder hacer invisibles o visibles personajes (o cualquier objeto 3D) durante un tiempo en concreto.
Entre otras cosas puede servir para visualizar selectores (durante unos segundos) en zonas del escenario, que indiquen al jugador que en ese punto hay que llevar a cabo una acción.

Se puede animar cualquier zona del escenario ('brushes' independientes o grupos de 'brushes').
Estas animaciones podrán dar comienzo desde el mismo instante que se inicia el nivel o podremos controlar su activación y reanudación durante el transcurso del juego y según las condiciones que establezcamos.
Esta característica ya estaba disponible en el Genesis3D original y en Entidad 3D v1.1, pero estaba sin documentar y sin la capacidad de poder controlar la animación durante el transcurso del juego.

Se puede variar ligeramente el comportamiento de los enemigos con una opción que hace que los enemigos no estén en continuo movimiento, sino que se detengan de vez en cuando. De todas formas si el enemigo está disparando al jugador, lo seguirá haciendo pero detenido en el sitio durante unos segundos. Esta característica utiliza por fin la animación 'Shoot' de los personajes, que es la animación del personaje disparando pero quieto en un sitio.

Nuevos comandos ('QVID' y 'DVID') para quitar y dar una vida al jugador.

La imagen de los créditos del juego de una resolución de 640x480, ahora se presenta a pantalla completa independientemente de la resolución de pantalla.

Nuevo comando ('FINJ') que establece el final real del juego, presentando una imagen y deteniendo el juego. El jugador sólo tendrá acceso al menú al pulsar 'ESC'.
La imagen que se presenta con este comando es de 640x480(color RGB) pero siempre se mostrará a pantalla completa independientemente de la resolución de pantalla.

El jugador no puede recoger munición de una arma que no tenga.

El tiempo en reaparecer los enemigos después de muertos, puede ser establecido independientemente para cada enemigo.

Nuevas características de Entidad 3D v2.1

A continuación se describen todas las nuevas características que se han implementado en la versión 2.1 con respecto a la versión 2.0:

Junto a Entidad 3D 2.1 se distribuyen las utilidades de Genesis3D que anteriormente había que instalar antes de instalar Entidad 3D.
Se cambia la carpeta por defecto de Entidad 3D, antes era 'GENESIS3D11' y ahora es 'entidad_3d'.
Todas las armas en formato .act ahora están en la carpeta 'armas', antes en la carpeta 'actors'.
Los objetos prefabricados de World Editor en formato .3dt ahora están en la carpeta 'prefabs', antes en la carpeta 'objects'.

Se actualizan diversas secciones del tutorial de World Editor y de Entidad 3D, también se añaden nuevas.

Se documenta en el tutorial de World Editor la utilización de objetos prefabricados 'prefabs' para utilizar como librería de objetos para World Editor.

Se corrige el error *'No se puede abrir el fichero'* que aparecía en ocasiones al utilizar dos comandos seguidos 'ACCI OPCI'.

Se añade la posibilidad de tener puntuación y récord en nuestro juego:

Se añade la posibilidad de comenzar a jugar a partir del último nivel del juego que alcanzó el jugador:

Se añade la posibilidad de que puertas y plataformas (entidades 'Door' y 'MovingPlat') tengan un movimiento ininterrumpido una vez han comenzado a moverse.

Algo parecido ya se estaba realizando hasta ahora con la entidad 'ModelControler' pero si bien el aspecto y movimiento de estos bloques era realista, no lo era tanto sus características físicas al interactuar con el jugador o con los enemigos, que en muchas ocasiones los atravesaban o en el caso de las plataformas no transportaban al jugador.
Esta nueva característica, entre otras cosas, permite:

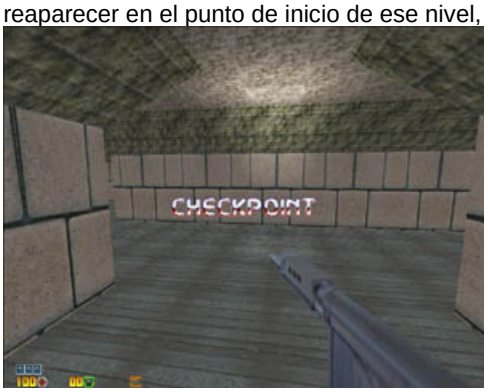
- Realizar juegos con plataformas móviles independientes que transporten realmente al jugador.
- Puertas o bloques que se mueven solos, el jugador debe esperar la secuencia correcta para pasar.
- Bloques de lava o sustancias radioactivas que se abalanzan sobre el jugador.
- Bloques que empujan al jugador hacia zonas con pinchos.
- Niveles con múltiples ascensores que el jugador deberá utilizar en la secuencia correcta para no caer al vacío.

Se mejora considerablemente la iluminación que se aplica al personaje protagonista:

Ahora se pueden añadir hasta 64 cámaras fijas por nivel. Con estas cámaras se podrán realizar las siguientes cosas:
Situarlas en cualquier lugar del nivel.
Activarlas y desactivarlas durante el juego mediante comandos de Entidad 3D.
Pueden activarse sólo durante un tiempo.
El juego, o un nivel en concreto, puede dar comienzo con la vista de una de estas cámaras.
Pueden apuntar continuamente hacia donde vaya el jugador:

Pueden quedarse completamente inmóviles independientemente de los movimientos del jugador:

Se añade la posibilidad de situar puntos de 'Checkpoint' en nuestro nivel, de esta forma al morir el jugador no tendrá que reaparecer en el punto de inicio de ese nivel, sino en el último 'Checkpoint' que haya alcanzado:



Se corrige la orientación de modelos 3D para las entidades 'SpecificActor' y 'ItemObject'. Ahora los ángulos introducidos en el parámetro 'orientation' se toman como grados (de 0 a 365) y no como radianes (de 0 a 6.2).
Esta modificación tiene implicaciones en los niveles creados con Entidad 3D 2.0 o inferior, por lo que se recomienda repasar los valores de 'Orientation' de todas las entidades 'SpecificActor' y 'ItemObject'.

Se puede variar, individualmente, la velocidad a la que se mueven los enemigos.

Se puede configurar, individualmente, si determinados enemigos disparan o no.

Se puede configurar, individualmente, si determinados enemigos quitan salud al jugador al tocarlos o no.

Se corrige el error que provocaba que Entidad 3D se quedase frenado en el logo inicial 'Genesis3D', en PC's con velocidades de 2.0Ghz o más.
También ha quedado solucionado el mismo problena en el programa 'Actor Viewer'.
Hasta ahora se resolvía, en Windows XP, estableciendo compatibilidad con Windows 98 en las propiedades de los programas.

Nuevas características de Entidad 3D v2.2

A continuación se describen las nuevas características que se han implementado hasta la versión 2.2i con respecto a la versión 2.1:

Se puede configurar si un enemigo comienza a moverse al ver al jugador o se mueve desde que comienza el nivel.
Entidad **'BotActorStart'**, parámetro **'Inicio_parado'**.

Se puede configurar el tiempo que un enemigo invierte en buscar al jugador una vez lo pierde de vista, transcurrido ese tiempo el enemigo se detiene y espera al jugador.
Entidad **'BotActorStart'**, parámetro **'Tiempo_corriendo'**.

Durante el juego se puede no permitir el movimiento lateral (derecha/izquierda) del jugador cuando por ejemplo esté controlando un vehículo sin movimiento lateral como pueden ser automóviles o motos.
Comandos **'LAT0'** y **'LAT1'**.
Entidad **'Configuracion'**, parámetro **'Movimiento_lateral'**.

Configuración del brillo del personaje protagonista para que se vea más claro o más oscuro en el juego, sin alterar para nada el fichero .ACT original del modelo 3D.
Entidad **'Configuracion'**, parámetro **'Brillo_jugador'**.

Configuración del salto que realiza el jugador al pulsar la tecla de saltar. Permitir saltar si o no y definición de la altura a la que saltará.
Entidad **'Configuracion'**, parámetro **'Salto_jugador'**.

Enemigos cobardes. Se pueden definir enemigos que siempre huyan del jugador.
Entidad **'BotActorStart'**, parámetro **'Cobarde'**.

Enemigos que se abalanzan. Se pueden definir enemigos que vayan directamente hacia el jugador, sin mantener una distancia de seguridad, para infringirle daño por el simple contacto. Ideal para zombies o fantasmas.
Entidad **'BotActorStart'**, parámetro **'Abalanzarse'**.

Configuración del área de colisión del personaje con las paredes.
Entidad **'Configuracion'**, parámetros **'Colisión_player..'**.

Posibilidad de cambiar, en pleno juego, cualquier textura del nivel.
Comando **'CTEX'**.

Posibilidad de incluir animaciones en el interior del escenario.
Entidad **'Animacion'**.
Comandos **'ANI1'** y **'ANI0'**.

Configuración del brillo de los enemigos para que se vean más claros o más oscuros en el juego, sin alterar para nada el fichero .ACT original del modelo 3D.
Entidad **'Configuracion'**, parámetros **'Brillo_enemigo_...'**.

Configuración del brillo de los objetos recogibles.
Entidad **'ItemObject'**, parámetro **'Brillo'**.

Configuración del brillo de los objetos o personajes estáticos.
Entidad **'SpecificActor'**, parámetro **'Brillo'**.

Se aplica el mismo nivel de brillo a las armas que al jugador.

Mejora en la vista de la muerte del jugador. Todas las cámaras en tercera persona ahora muestran al jugador cayendo, no como antes que se veía siempre esta acción en primera persona.

Posibilidad de encender y apagar luces durante el juego.
Entidad **'DynamicLight'**, parámetros **'Num_luz'** y **'Visible'**.
Comandos **'LUZ1'** y **'LUZ0'**.
La entidad 'DynamicLight' se incorpora al manual de Entidad 3D, antes estaba parcialmente explicada en el manual de 'World Editor'.

Posibilidad de encender y apagar efectos corona durante el juego.
Entidad **'Corona'**, parámetros **'Num_corona'** y **'Visible'**.
Comandos **'COR1'** y **'COR0'**.
La entidad 'Corona' se incorpora al manual de Entidad 3D, antes estaba parcialmente explicada en el manual de 'World Editor'.

Soporte de texturas animadas en los niveles creados con 'World Editor' para Entidad 3D.

Comandos **'TEA1'** y **'TEA0'**.
El funcionamiento de las texturas animadas se explica en un apartado de la sección 'Anexos' del manual 'on-line' de Entidad 3D.

Se pueden procesar comandos al recoger objetos.
Entidad **'ItemObject'**, parámetro **'CommandsFile'**.

Nueva cámara configurable llamada 'Plataformas'.
Entidad **'Configuracion'**, parámetros **'Camara_inicial_PLAT'**, **'Camara_PLAT_alt'**, **'Camara_PLAT_dist'** y **'Permite_camara_PLAT'.**

Nuevo control del personaje protagonista. Con cuatro teclas (o 4 botones del Joystick) se puede controlar el personaje sin utilizar el ratón.
Entidad **'Configuracion'**, parámetro **'Avance_direccion'**.

Permite seleccionar y utilizar la animación de salto del personaje protagonista.
Entidad **'Configuracion'**, parámetro **'Anim_salto'**.

(Para más información sobre estas nuevas características, consulta el manual 'on-line' de Entidad 3D).

Nuevas características de Entidad 3D v2.3

A continuación se describen las nuevas características que se han implementado en la versión 2.3a con respecto a la versión 2.2i:
Arma número 1 completamente configurable. Se incluyen 37 nuevas armas ya preconfiguradas.
Entidad **'ConfigArmaUno'**.

Diferenciación del sexo del enemigo para sonidos de daño y muerte.
Entidad **'BotActorStart'**, parámetro **'Masculino'**.

Corrección de alineación y orientación de las otras 3 armas que aún no son configurables (Arma número 2, arma número 3 y arma número 4).

Se incrementa el ángulo de disparo que se permite hacia arriba y hacia abajo en primera persona.

Animación de retroceso del arma del jugador al disparar en vistas en tercera persona.

Mejora de la colisión del personaje y enemigos en la zona de la cabeza.

Se incluye un nivel de ejemplo llamado 'tiro_al_pato', diseñado para probar las nuevas armas que puedas crear o las que ya se incluyen por defecto.

(Para más información sobre estas nuevas características, consulta el manual 'on-line' de Entidad 3D).

Nuevas características de Entidad 3D v2.4

A continuación se describen las nuevas características que se han implementado en la versión 2.4 con respecto a la versión 2.3a:

El jugador puede avanzar agachado. Tecla 'C' se agacha, tecla 'V' se levanta.
Entidad **'DeathMatchStart'**, parámetros **'Anim_agachado_Idle'** y **'Anim_agachado_Walk'**

Traspaso de algunos parámetros de la entidad 'Configuracion' a la entidad 'DeathMatchStart', con el fin de unificar los parámetros que tienen que ver directamente con el jugador.
Los parámetros movidos son:

Anim_salto
Vidas
Jugador_masculino
Salto_jugador
Brillo_jugador
Movimiento_lateral
Colision_player1p
Colision_player1n
Colision_player2p
Colision_player2n
Sonido_pasos
Pierde_armas
Ver_arma
Usar_Blaster

Aparece explosión de sangre en enemigos por impactos de arma número 1 o número 4.
Entidad **'BotActorStart'**, parámetro **'Sangre'**.

Arma número dos completamente configurable.
Esta arma ahora también permite 'plantar' minas antipersona que sólo explotan al contacto con enemigos o con el propio jugador. La mina permanece en el suelo hasta que la pise alguien.
Nueva entidad **'ConfigArmaDos'**.

Arma número tres completamente configurable.
Nueva entidad **'ConfigArmaTres'**.

Arma número cuatro completamente configurable.
Nueva entidad **'ConfigArmaCuatro'**.

Permite definir hasta 10 tipos de sonidos distintos de muerte para enemigos (antes sólo había uno):
Se incluyen 10 ficheros de sonido .WAV de muestra listos para usar en tu juego.
Nueva entidad **'ConfigSonidosEne'**.

Permite definir hasta 10 tipos de sonidos distintos de muerte para enemigas (antes sólo había uno):
Se incluyen 10 ficheros de sonido .WAV de muestra listos para usar en tu juego.
Nueva entidad **'ConfigSonidosEne'**.

Sonidos de daño diferenciados para enemigos y jugador:
4 sonidos distintos para enemigos
4 sonidos distintos para enemigas
5 sonidos distintos para jugador
5 sonidos distintos para jugadora
Se incluyen los ficheros de sonido .WAV correspondientes listos para usar en tu juego.
Nueva entidad **'ConfigSonidosEne'**.
Nueva entidad **'ConfigSonidosJug'**.

Sonidos de muerte diferenciados para enemigos y jugador:
10 sonidos distintos para enemigos
10 sonidos distintos para enemigas
5 sonidos distintos para jugador
5 sonidos distintos para jugadora
Se incluyen los ficheros de sonido .WAV correspondientes listos para usar en tu juego.
Nueva entidad **'ConfigSonidosEne'**.
Nueva entidad **'ConfigSonidosJug'**.

Los enemigos pueden emitir sonidos.
Puede utilizarse para que insulten al jugador, para llamar su atención o en el caso de monstruos para que simplemente emitan un sonido constante amenazante.
Estos sonidos son posicionales y se producen en la posición exacta del enemigo que los emite por lo que el jugador los oye en el lado correspondiente y con la atenuación adecuada en función de la distancia.
Pueden ser dos sonidos distintos por enemigo para no hacer muy monótono un solo sonido por enemigo.
Se incluyen los ficheros de sonido .WAV correspondientes (con voces en castellano) listos para usar en tu juego.
Nueva entidad **'ConfigSonidosEne'**.
Entidad **'BotActorStart'**, parámetro **'Tiempo_sonido'**.

El nuevo comando 'QSAL' permite quitar una determinada cantidad de salud al jugador en cualquier situación de tu juego:
Nuevo comando **'QSAL'**.

Se permite hacer que el jugador atraviese los modelos 3D (ficheros .ACT) colocados en el escenario con la entidad 'SpecificActor'.
Esto ya se podía hacer retocando los valores Bloqueo_X, Bloqueo_Y, Bloqueo_Z, pero ahora será más fácil si lo que queremos es que no haya una caja invisible que colisione con el jugador. Esto también permitirá que sea, en World Editor, donde definamos las zonas de colisión de ese modelo 3D puesto con 'SpecificActor', obteniendo de esta manera resultados más realistas.

Entidad **'SpecificActor'**, parámetro **'Bloqueo'**.

(El podio donde está subido el jugador es un fichero .ACT)

Dos nuevos comandos permiten hacer que el jugador no se pueda desplazar durante un tiempo dado o indefinidamente. Comandos **'MOV0'** y **'MOV1'**.

(Para más información sobre estas nuevas características, consulta el manual 'on-line' de Entidad 3D).

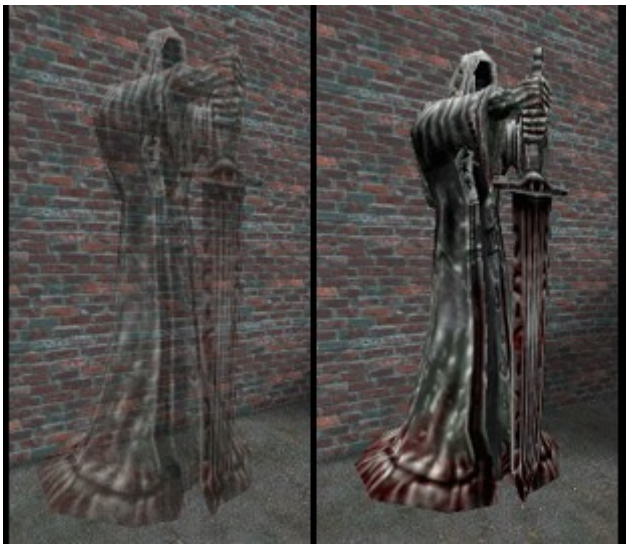
Nuevas características de Entidad 3D v3.0

A continuación se describen las nuevas características que se han implementado en la **versión 3.0** con respecto a la versión 2.4:

Diversas mejoras en la entidad 'ItemHealth':
Incremento de nivel salud variable
Hasta tres tipos de objetos diferentes al mismo tiempo para representar los 'kits' de salud
Brillo del objeto variable
Nivel de transparencia del objeto variable
Tamaño del objeto variable

Diversas mejoras en la entidad 'ItemVida':
Brillo del objeto variable
Nivel de transparencia del objeto variable
Tamaño del objeto variable

Diversas mejoras en la entidad 'ItemArmor':
Incremento de nivel armadura variable
Brillo del objeto variable
Nivel de transparencia del objeto variable
Tamaño del objeto variable



En la entidad 'SpecificActor':
Nivel de transparencia del objeto/personaje variable

En la entidad 'Configuracion':
Nuevo parámetro 'Ver_Sombras'

Corrección en error de orientación por defecto del arma número dos

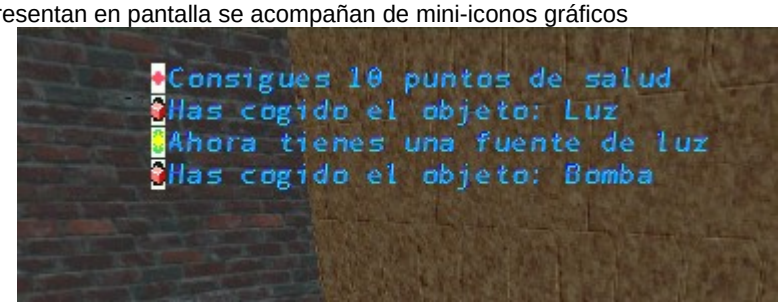
Nuevo comando 'DSAL' para dar determinados niveles de salud al jugador

Presenta un mensaje al no poder recoger objetos no recogibles

Corregido el error que no quitaba la fuente de luz del inventario del jugador si se utilizaba el comando 'QINV'

Corregido el error que reproducía la animación de saltar del jugador cuando bajaba por ascensores

Los mensajes que se presentan en pantalla se acompañan de mini-iconos gráficos



Panel con la lista de objetivos de la misión:
Nueva entidad llamada 'Objetivo'
Nuevo comando 'PAN1' para añadir un nuevo objetivo al panel durante el juego
Nuevo comando 'PAN0' para quitar un objetivo del panel durante el juego
Nuevo comando 'OBJ1' para establecer objetivos cumplidos durante el juego

Modificaciones en la entidad 'ConfigArmaCuatro' para que soporte armas cuerpo a cuerpo:
Nuevo parámetro 'Alcance' (alcance del disparo)
Nuevo parámetro 'AnimDisparo' (nombre de la animación de ataque o disparo)
Nuevo parámetro 'AnimDispVel' (velocidad de la animación anterior)
Nuevo parámetro 'ArmaDeMano' (no se descuenta munición)

Se han corregido dos errores en el programa que evitan fallos que aparecían en algunos PC y que no permitían iniciar el programa.

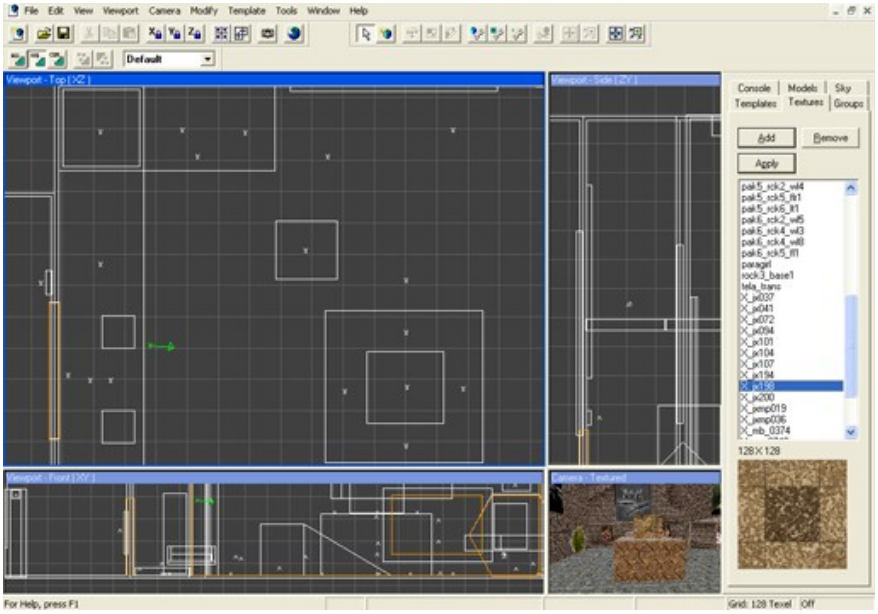
Diversas mejoras en la entidad 'BotActorStart' para modificar el comportamiento de los enemigos y para poder usar esta entidad para crear personajes que no sean enemigos y que puedan interactuar con el jugador:
Nuevo parámetro 'CobardePorc' (porcentaje de tiempo en modo cobarde)
Nuevo parámetro 'Indiferente' (enemigo dando vueltas)
Nuevo parámetro 'IndiferentePorc' (porcentaje de tiempo en modo indiferente)
Nuevo parámetro 'Buscar' (enemigo busca al jugador o no)
Nuevo parámetro 'BuscarPorc' (porcentaje de tiempo que busca al jugador)
Nuevo parámetro 'DistanciaMin' (Distancia mínima entre enemigo y jugador)
Nuevo parámetro 'AnimacionTipo' (Para seleccionar andar o correr)
Nuevo parámetro 'Bloqueo' (Para detectar colisión de enemigo con jugador)
Nuevo parámetro 'Comandos_colision' (Para indicar un fichero de comandos .INI para procesar cuando el enemigo colisione con el jugador)
Nuevo parámetro 'Scale' (Para modificar el tamaño de los enemigos)
Los enemigos que no disparan no encaran al jugador, a menos que sean estáticos
La pérdida de salud del jugador por contacto con enemigos se realiza más lentamente

Nueva entidad 'SunLight' para crear luz de Sol realista al producir rayos de luz siempre paralelos, independientemente de la distancia que haya entre la entidad y los objetos.

Nuevo comando 'INPU' para que el jugador pueda entrar textos en pleno juego, y una vez introducidos los podamos procesar.

Modificaciones en las entidades 'ConfigArmaUno', 'ConfigArmaDos', 'ConfigArmaTres' y 'ConfigArmaCuatro' para poner animar el arma que lleve el jugador
El arma del jugador podrá tener una animación en primera persona mientras la lleva sin disparar o atacar. También podrá tener otra animación diferente mientras dispara o ataca:
Nuevo parámetro 'AnimDisparo' (Animación al disparar)
Nuevo parámetro 'AnimDispVel' (Velocidad de la anterior animación)
Nuevo parámetro 'AnimNDisparo' (Animación sin disparar)
Nuevo parámetro 'AnimNDispVel' (Velocidad de la anterior animación)
Nuevo parámetro 'Municion_resta' (Cantidad de munición a descontar en cada disparo)

A modo de prueba, en esta distribución de Entidad 3D, se incluye la versión 2.0 Beta 2 del editor de niveles World Editor de Michael R. Brumm.



Se sigue incluyendo también en esta distribución la antigua versión de World Editor (la 1.0) y se mantendrá como principal editor, pero he incluido también esta nueva versión por ser completamente compatible con los niveles y texturas creados con la anterior versión, y para que dispongamos de una segunda alternativa para la creación de niveles para Entidad 3D.

Para conocer algunos detalles sobre este editor, consulta el manual de Entidad 3D, apartado **World Editor 2.0** del índice principal.

Modificaciones en la entidad 'ConfigArmaUno' para que el arma número 1 pueda tener fogonazo al disparar:
- Nuevo parámetro 'Escala_fogonazo_1'
- Nuevo parámetro 'Escala_fogonazo_3'
- Nuevo parámetro 'Fogonazo_BMP'
- Nuevo parámetro 'Fogonazo_aBMP'

Se incluyen nuevas armas que hacen uso de las nuevas características de animación de armas y armas cuerpo a cuerpo. Esto lo podrás ver consultado las nuevas entidades 'ConfigArmaUno', 'ConfigArmaDos', 'ConfigArmaTres' y 'ConfigArmaCuatro' que se incluyen en el nivel de ejemplo 'Tiro al pato'.

La tecla 'P' pausa y reanuda el programa. Antes se utilizaba la tecla 'P' para pausa y la tecla 'L' para reanudar.

Corrección de error que no visualizaba la animación de la entidad 'Animacion' al reiniciar el nivel, ya fuera por volver a comenzar la partida o por cambio de resolución durante el juego.

Modificaciones en la entidad 'ConfigSonidosEne' para configurar sonidos personalizados para cada uno de los enemigos:
- 2 sonidos mientras corre o camina
- 3 sonidos para daño
- 3 sonidos para muerte

Evita que el arma se vea sin el personaje cuando la cámara entra en un bloque sólido.

Corrección de error que no detectaba colisiones con modelos de 'SpecificActor'.

Nueva sección en el nivel de ejemplo 'Tiro al pato' que representa una ciudad:

Esta ciudad te aparecerá de día, pero puedes hacer que aparezca de noche siguiendo estos puntos:
En la entidad 'SunLight' pon el parámetro 'light' a 1.
En la pestaña 'Sky' cambia los 6 nombres de texturas de un cielo de día por los siguientes 6 nombres de texturas que representan el cielo de noche, y en este orden:
noche_bk
noche_ft
noche_up
noche_dn
noche_rt
noche_lf
Compila el nivel usando una luz ambiental 'Default Light Level (RGB)' 70.0 70.0 70.0
Ya dentro del nivel puedes acabar de crear unos efectos de corona de luz en las farolas dirigiéndote a la zona donde se encuentra un depósito con un teclado y una pantalla, en ese punto pulsa 'Acción' (tecla 'Espacio').

Posibilidad de cargar el juego sin necesidad de crear accesos directos.
El jugador comenzará a jugar a nuestro juego con solo ejecutar en programa 'entidad3d.exe'.
Para saber como utilizar esta característica consulta el manual de Entidad 3D, se explica en el apartado '**Preparando un acceso directo**'.

Esta nueva versión de Entidad 3D incluye una actualización del motor 3D (Genesis3D) gracias a la recopilación de mejoras que a realizado Quest of Dreams.
Esta actualización del motor 3D implica:
Aumento de la velocidad en un 40% aproximadamente.
Resoluciones de pantalla de hasta 1280x1024 32bpp. Para activar los modos de 24 y 32 Bpp, consulta el manual de Entidad 3D, apartado '**Calidad del color y OpenGL**'.
Escenarios con texturas de 24 bits de color. Consulta esta característica en el manual de World Editor, apartados '**Añadiendo nuevas texturas**' y '**Texturas con transparencias**'.
En los modos de pantalla de 32bpp conseguiremos escenarios sin defectos en texturas en movimiento, tampoco se producirán parpadeos en transparencias.
Nueva versión del programa Texture Packer que soporta texturas de 24 bits.
Nuevos 'drivers' de pantalla para manejar Direct 3D.
A modo experimental también se incluirá un 'driver' para OpenGL. Consulta esta característica en el manual de Entidad 3D, apartado '**Calidad del color y OpenGL**'
Para usuarios avanzados, el nuevo motor 3D les permite configurar diversos parámetros de funcionamiento de Direct 3D como: Bpp, ZBufferD, Back Buffer, CompressTextures, FSAntiAliasing, ExtraTextures, NoVsync, Async, ...
Consulta esta característica en el manual de Entidad 3D, apartado '**Calidad del color y OpenGL**' o leyendo los comentarios que hay en el fichero '**D3D24.ini**' que se encuentra en la carpeta principal de Entidad 3D.

Consideraciones a tener en cuenta para adaptar niveles creados con una versión de Entidad 3D anterior a la 3.0:
Modifica todas las texturas con transparencias que utilice tu nivel. Tendrás que convertir las texturas de 256 colores (con el último color transparente) a texturas de 24 bits y como color transparente el R:255 G:0 B:254
En los 'brushes' que uses como puertas que se abren en bisagra (como una puerta normal de una casa), cambia el tipo de 'brush' de 'Solid' a 'Window', de lo contrario el personaje tendrá muchos problemas para pasar.

Para más información sobre estas nuevas características, consulta las secciones de comandos y entidades de los manuales de Entidad 3D y World Editor

Nuevas características de Entidad 3D v3.2

A continuación se describen las nuevas características que se han implementado en la versión 3.2 con respecto a la versión 3.0:

Corrige un error que aparecía en algunos PCs al iniciar determinadas resoluciones de pantalla.

Soporte para los siguientes formatos gráficos gracias a la librería FreeImage:
BMP, JPG, GIF, LBM, IFF, PBM, PCX, PGM, PNG, TGA, TIF, PSD, ICO
... para utilizar en Entidad 3D únicamente para:
- Imágenes iniciales de presentación del juego
- Imagen de portada antes de la carga del nivel
- Imagen mientras se carga el nivel
- Imagen de créditos del juego
- Imagen para final del juego

Corrige un error que aparecía al lanzar los niveles usando '**autonivel.ini**'

Permite fijar rutas preestablecidas para los enemigos o personajes.
- Nueva entidad '**Posicion**'.
- Nuevos parámetros '**Destino1**', '**Destino2**', '**Destino3**', '**Destino4**' y '**Destino5**' de la entidad '**BotActorStart**'.

Permite establecer niveles de inmunidad para enemigos o personajes.
- Nuevo parámetro '**Inmune**' de la entidad '**BotActorStart**'.

El arma número 1 deja de tener munición infinita para el jugador, ahora consume munición a medida que se dispara.
Se puede volver a configurar esta arma con munición infinita con sólo poner 0 en el parámetro 'Municion_resta' de la entidad 'ConfigArmaUno', al igual que ya se podía hacer con el resto de armas.

El arma número 1 también deja de tener munición infinita para los enemigos, ahora se puede configurar como el resto de armas:
En la entidad '**BotActorStart**' nuevos parámetros '**WeaponBlaster**' y '**WeaponBlasterA**'

Nueva entidad '**ItemBlasterAmmo**' que representará la munición que se puede recoger en el escenario para el arma número 1.

Para las cuatro armas se puede establecer la cantidad de munición inicial del arma al recogerla por primera vez y la cantidad de munición de los cargadores de cada arma:
- En la entidad '**ConfigArmaUno**' nuevos parámetros '**Municion_inicial**' y '**Municion_cargador**'
- En la entidad '**ConfigArmaDos**' nuevos parámetros '**Municion_inicial**' y '**Municion_cargador**'
- En la entidad '**ConfigArmaTres**' nuevos parámetros '**Municion_inicial**' y '**Municion_cargador**'
- En la entidad '**ConfigArmaCuatro**' nuevos parámetros '**Municion_inicial**' y '**Municion_cargador**'

Corrige error que no aplicaba el parámetro '**Cadencia**' de las entidades '**ConfigArma...**', a las armas de los enemigos.

Corrige error que dejaba a un enemigo inactivo si el jugador moría al colisionar con él.

Para enemigos, permite configurar las distancias mínimas a las que debe estar el jugador para utilizar cada arma.
En caso de un arma cuerpo a cuerpo se puede utilizar para que el enemigo sólo ataque al estar cerca del jugador.
Estas distancias las podemos establecer en la misma entidad '**BotActorStart**' del enemigo en los parámetros '**Distancia1_disparar**', '**Distancia2_disparar**', '**Distancia3_disparar**' y '**Distancia4_disparar**' para fijar las distancias mínimas para utilizar cada arma.

Permite que cuando los enemigos están en modo ataque no realicen movimientos laterales aleatorios.
En la entidad '**BotActorStart**' nuevo parámetro '**Mov_lateral_azar**'

Para enemigos, permite establecer que al disparar el arma siempre lo haga parado y reproduciendo la animación 'Shoot' de ese enemigo.
Se puede configurar independientemente para las 4 armas que lleve el enemigo, pero esta característica tiene una mayor utilidad para el arma 4 cuando es un arma cuerpo a cuerpo, ya que de esta forma el enemigo reproducirá la animación de ataque 'Shoot' sin correr, causando daño cuando esté cerca del jugador, siendo ésta animación el enemigo usando una espada, cuchillo, palo, palanca o lo que se nos ocurra diseñar.
En la entidad '**BotActorStart**' nuevos parámetros '**Dispara1_parado**', '**Dispara2_parado**', '**Dispara3_parado**' y '**Dispara4_parado**'

Se añade opción para que los objetos recogidos por el jugador se añadan al inventario de objetos o no.
En la entidad '**ItemObject**' nuevo parámetro '**Inventario_objeto**'

Permite variar el tiempo que tarda en reaparecer cada objeto de forma independiente.

En la entidad '**ItemObject**' nuevo parámetro '**Tiempo_reaparicion**'

Mejora en la explosión de la sangre por impactos recibidos de arma 4.
Ahora está un poco más dispersa y más cerca de la víctima. En primera persona el jugador que recibe los impactos ve que se le llena la pantalla con la explosión de su sangre.

Se añade opción para que aparezca una charco de sangre debajo de los enemigos muertos y también debajo del personaje del jugador muerto.

En la entidad '**Configuracion**' nuevos parámetros '**Sangre_charco_1** a **Sangre_charco_8**' para enemigos.
En la entidad '**DeathMatchStart**' nuevo parámetro '**Sangre_charco**' para jugador.

Se incorpora al manual de Entidad 3D las siguientes entidades que sólo aparecían parcialmente comentadas en el tutorial de World Editor:
'**ItemGrenade**', '**ItemGrenadeAmmo**'
'**ItemRocket**', '**ItemRocketAmmo**'
'**ItemShredder**', '**ItemShredderAmmo**'
'**ChangeLevel**'

No se reponen las vidas del jugador al pasar al siguiente nivel.

La puntuación no se pierde al pasar al siguiente nivel.

El autoguardado de la partida al finalizar cada nivel ahora también guarda la puntuación y las vidas del jugador.

Los enemigos ya no hacen saltar al jugador al siguiente nivel al pisar un 'model' asociado a la entidad 'ChangeLevel'.

Se añade la posibilidad que al cargar un nuevo nivel se ejecute un fichero de comandos .INI justo al iniciar el nivel.
En la entidad '**ChangeLevel**' nuevo parámetro '**CommandsFile**'

Nueva entidad '**Fx_Surtidor**' que permite definir efectos de partículas. Con esta entidad se pueden crear efectos de fuego, humo, vapor, chispas, gases, etc...
Esta nueva característica de Entidad 3D está basada en el código fuente original de 'David Wulff'.

Nueva entidad '**Fx_Lluvia**' que permite añadir lluvia y nieve a los niveles.
Esta nueva característica de Entidad 3D está basada en el código fuente original de 'David Wulff'.

Nueva entidad '**FX_Sprite**' que permite crear partículas en movimiento en zonas concretas.
Esta nueva característica de Entidad 3D está parcialmente basada en el código fuente original de 'David Wulff'.

Nueva entidad '**FX_Niebla**' que permite añadir un efecto de niebla de fondo en los niveles.

Nuevos comandos '**NIE1**' y '**NIE0**' que permiten activar y desactivar la niebla de fondo durante el transcurso del juego.

Nuevos comandos '**SODF**', '**SODP**' y '**SODR**' que permiten añadir y controlar un segundo canal de sonido cíclico de fondo.

Cuando los enemigos tenían activado '**Mov_lateral_azar**' iban directos hacia el jugador y si lo perdían de vista o chocaban con una pared que les impedía ver al jugador, buscaban como esquivar la pared para volver a ver al jugador. Pero si chocaban con una pared baja que les impedía llegar al jugador pero lo tenían a la vista, no intentaban esquivar esa pared. Esto ha sido resuelto.

Ahora el programa Entidad 3D dispone de un sistema de seguimiento de procesos para consultar en caso de que el programa falle sin presentar ningún error en pantalla. Estos procesos se almacenan en el fichero '**Informe_Entidad_3D.Log**' en la carpeta principal de Entidad 3D. En caso de que el programa Entidad 3D falle sin poder presentar un mensaje de error, al menos se podrá consultar ese fichero para saber que es lo que el programa estaba haciendo o se disponía a hacer antes de producirse el error.

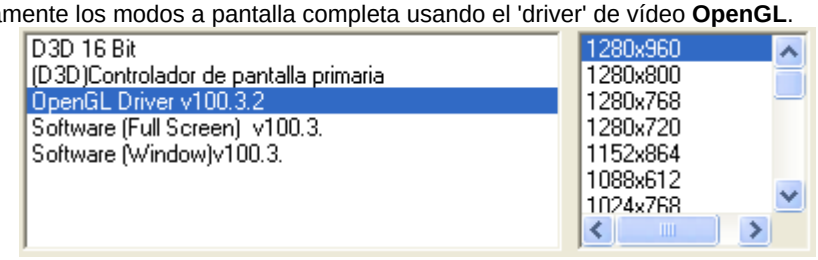
Para más información sobre estas nuevas características, consulta las secciones de los comandos y las entidades correspondientes que encontrarás en este mismo manual.

Nuevas características de Entidad 3D v3.3

A continuación se describen las nuevas características que se han implementado en la versión 3.3 con respecto a la versión 3.2:

Soluciona error que al usar el comando '**PINV**' provocaba un error en el programa y no añadía objetos al inventario del jugador.

Ahora se ven correctamente los modos a pantalla completa usando el 'driver' de vídeo **OpenGL**.



Posibilidad de que al pulsar una tecla se lance un fichero de comandos .INI durante el transcurso del juego. (16 teclas como máximo).
Nueva entidad 'TeclaComandos'.

Nueva entidad 'ConfigEnemigos' con parámetros generales de los enemigos que anteriormente se establecían en la entidad 'Configuracion'.
Los parámetros movidos a esta nueva entidad son:

Brillo_enemigo_1	Sangre_charco_1
Brillo_enemigo_2	Sangre_charco_2
Brillo_enemigo_3	Sangre_charco_3
Brillo_enemigo_4	Sangre_charco_4
Brillo_enemigo_5	Sangre_charco_5
Brillo_enemigo_6	Sangre_charco_6
Brillo_enemigo_7	Sangre_charco_7
Brillo_enemigo_8	Sangre_charco_8

Los parámetros establecidos con una entidad 'ConfigEnemigos' situada en un nivel, seguirán siendo utilizados en los siguientes niveles sin necesidad de poner una entidad 'ConfigEnemigos' en cada nivel.
Al estar ahora estos parámetros separados se puede usar esta nueva entidad para establecer diferentes parámetros generales para los enemigos en función del nivel que situemos esta entidad.

Permite poner enemigos/personajes transparentes mediante la nueva entidad 'ConfigEnemigos'.

Permite establecer la cantidad de salud que resta la lava al tocarla.
En entidad 'Configuracion' nuevo parámetro '**Lava_salud**'

Posibilidad de lanzar un fichero de comandos .INI al dar comienzo el primer nivel del juego.
En entidad 'Configuracion' nuevo parámetro '**FicheroComandos**'

Hasta ahora sólo podía haber 8 enemigos/personajes distintos para todo el juego y todos los niveles. Ahora esos 8 enemigos/personajes podrán ser distintos en cada nivel.

El comando 'TEXT' ahora permite presentar en pantalla variables, que pueden ir mezcladas con textos.
Puede servir por ejemplo para presentar al jugador el dinero que pueda llevar, un nivel de energía, un nivel de poder mágico, fuerza, número de tarjetas de acceso, bolas mágicas que le faltan por recoger, etc...

Diversas mejoras en operaciones con variables:
Nuevo comando 'OPEC' para añadir una cadena de texto a una variable.
Nuevo comando 'OPE*' para operaciones de multiplicación con una variable.
Nuevo comando 'OPE/' para operaciones de división con una variable.
Ahora todos los comandos que se utilizan para operaciones con variables soportan otra variable como segundo operando.
Los comandos de control 'IF COND' ahora también soportan comparaciones con otra variable.

Nuevo sistema de mensajes, con 55 tipos de letra diferentes, que permite situar mensajes y datos en cualquier posición de la pantalla:
Nuevos comandos 'MSJ1' y 'MSJ0'.

Se corrige error que producía unas líneas negras en los márgenes inferior y derecho de la imagen y que se notaban más cuando se utilizaba niebla de fondo al aparecer líneas blancas. Este defecto también quedaba reflejado en las capturas de pantalla.

Ahora con el uso de la niebla de fondo se pueden conseguir rendimientos de velocidad muy altos al tener la posibilidad de que el motor 3D no calcule nada de lo que hay por detrás de la niebla.
En entidad 'Fx_Niebla' nuevo parámetro '**RenderTrasNiebla**'

Se añaden 2 líneas más a los textos que presenta simultáneamente el comando 'TEXT' en la consola superior (ahora 6 líneas en total). También se incrementa el tiempo que permanecen los textos en pantalla.

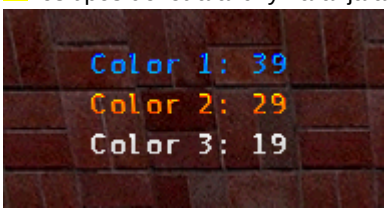
Nuevo sistema de superposición de imágenes 2D en pantalla.
Hasta ahora esto se podía realizar con el comando 'IMG1' pero sólo con 5 imágenes simultáneamente en pantalla y todas ellas compartiendo la misma paleta gráfica de 256 colores.
Ahora se podrán superponer hasta 64 imágenes simultáneamente y cada una de ellas con su propia paleta gráfica de 256 colores o bien usar imágenes con color RGB, ambos tipos de imágenes con transparencias.
Esto, permite definir todo tipo de marcadores, indicadores, mensajes gráficos, ventanas, cuadros, etc.. para personalizar más nuestro juego.
Nuevos comandos 'OVERLAY1' y 'OVERLAY0'.

Ahora las cámaras 'Plataformas' y 'RPG' seguirán mostrando a los enemigos y al jugador aunque estas cámaras choquen o atraviesen bloques sólidos del escenario.
En el caso de la cámara 'Plataformas' sólo funcionará si esta cámara se establece como la cámara inicial fijando 'True' en el parámetro 'Camara_inicial_PLAT' de la entidad 'Configuracion'.
En el caso de la cámara 'RPG' funcionará si esta cámara se establece como la cámara inicial fijando 'True' en el parámetro 'Camara_inicial_RPG' de la entidad 'Configuracion'.

Si el jugador dispone de armadura sólo se le restará la mitad de salud que se le restaría sin armadura.

Se añade la posibilidad de situar los elementos del marcador (vidas, salud, armadura, armas, inventario, puntuación y récord) en cualquier posición de la pantalla y por separado. También se pueden ocultar los elementos que se seleccionen.
Nueva entidad 'Marcador'
En la entidad 'Configuracion' se elimina el parámetro '**ver_puntuacion**'

Los tipos de letra azul y naranja ahora se pueden usar para situar textos o variables en cualquier lugar de la pantalla. Para esta función se añade un tercer tipo de letra de color blanco.
Mediante los nuevos comandos 'MSJ1' y 'MSJ0'.



Redefinición completa de las teclas.
La opción del menú del jugador 'Configurar controles' permite modificar todas las teclas que se usan en el juego.

Nuevo inventario de objetos, con nuevo diseño, más capacidad y selector de objeto. Este inventario se podrá ocultar o hacer aparecer pulsando una tecla (tecla '**i**' por defecto).

Se documenta el comando 'EXIT' que en realidad existe desde las primeras versiones de Entidad 3D.

El programa Entidad 3D, el motor 3D Genesis3D, los drivers de pantalla y el motor de creación de los escenarios con World Editor, han sido recompilados con una nueva versión del compilador MS Visual C++ que utilizo para programar Entidad 3D. El resultado ha sido un aumento general de la velocidad del programa Entidad 3D.
Después de diversos test de velocidad los resultados son los que ves a continuación, aunque evidentemente son datos obtenidos en una sola máquina que presenta esta configuración: Intel Pentium IV 540 HT 3.2Ghz, 1GB RAM, MS Windows XP SP2, MS DirectX 9.0c, T.Gráfica nVidia GForce 6600GT PCI Express con 'drivers' DIC/2005:
Entidad 3D usando 'driver' de pantalla (D3D)Controlador de pantalla primaria:
Aumento de la velocidad en un 60%

Entidad 3D usando 'driver' de pantalla OpenGL:
Aumento de la velocidad en un 55%

Entidad 3D usando 'driver' de pantalla D3D 16 Bit:
Aumento de la velocidad en un 10%

World Editor 1.0 y 2.0:
Aumento de la velocidad de compilación de los niveles en un 40%

Para más información sobre estas nuevas características, consulta las secciones de los comandos y las entidades correspondientes que encontrarás en este mismo manual.

Nuevas características de Entidad 3D v3.4

A continuación se describen las nuevas características que se han implementado en la versión 3.4 con respecto a la versión 3.3:

Algunas mejoras en el procesamiento de los ficheros de comandos .INI:
Cuando se utilizaban entidades con procesamiento de comandos utilizando el botón 'Acción' y se colocaban muy juntas, sólo se procesaba uno de los ficheros de comandos, ahora se procesan todos los que estén en el radio de acción del jugador.
Utilizando entidades 'SpecificActor' para procesar ficheros de comandos al colisionar el jugador con estas entidades, sólo se procesaban los comandos cuando el jugador se encontraba en la posición inicial del 'SpecificActor'. Esto es correcto si se utiliza un modelo .ACT que no tiene desplazamiento, pero cuando se utiliza uno con desplazamiento, por ejemplo un pez dando vueltas en círculo, cuando el pez alcanzaba al jugador no se procesaban los comandos, aunque la colisión si que se detectaba correctamente y el jugador sufría un empuje. Esto ha sido corregido y ya se procesan los comandos al colisionar el jugador con un 'SpecificActor' que esté en movimiento.

Posibilidad de usar acciones combinadas con objetos del inventario y procesar ficheros de comandos .INI en función de las combinaciones de accion+objeto que se deseen.
Estas combinaciones se pueden controlar mediante comandos, por lo que podrán establecerse a través de las entidades que ya soportan ficheros de comandos .INI. También existe la posibilidad de controlar combinaciones que pueda hacer el jugador en cualquier lugar y momento del juego.
Se pueden usar las acciones que hay por defecto, modificarlas o añadir nuevas acciones:
Nueva entidad AccionMenu
Nuevas variables del sistema '**ACCION_SELECCIONADA** y '**OBJETO_SELECCIONADO**'

'Bugs' solucionados:
- Selección correcta de objetos en el inventario del jugador.
- Permite 50 entidades 'ItemObject' por nivel.
- Liberación de memoria correcta para imágenes de presentación.
- El jugador se quedaba inmóvil al saltar de nivel habiendo usado previamente el comando 'MOV0'.
- Se reduce el consumo de memoria entre niveles.

Hasta ahora sólo se permitían 36 entidades 'ItemObject' por nivel, aunque debido a un 'bug' del programa sólo cargaba 36 en todo el juego.
Ahora se permiten 50 entidades '**ItemObject**' por nivel y el 'bug' está solucionado.

Inventario de objetos 3D.
Ahora el inventario de objetos presenta el objeto seleccionado correspondiente en 3D y dando vueltas.
Sólo funciona en objetos recogidos por el jugador como entidades 'ItemObject' y que no reaparezcan u objetos entregados con el comando 'PINV' de los cuales el jugador ya tenga alguno igual en el inventario que recogió previamente como entidad 'ItemObject'.
En primera persona aparece enfrente del jugador, mientras que con las cámaras en tercera persona el objeto aparece encima del jugador.
Esta nueva característica del inventario es opcional y se puede activar o desactivar mediante el nuevo parámetro '**Inventario_3D**' de la entidad DeathMatchStart.

Con las entidades 'Command', que son invisibles, ahora existe la posibilidad de que se puedan ver en el escenario en forma de una pequeña bola semi-transparente.

Durante el desarrollo del juego, esto permite identificar las posiciones exactas de las entidades 'Command' cuando se juega a un nivel.
Nuevo parámetro '**Visible**' en la entidad Command.

El comando TEM1 ahora soporta hasta 15 temporizadores simultáneos.

Se implementa desgaste en los objetos del inventario.
El inventario de objetos muestra en el interior del icono de cada objeto el número de usos, de ese objeto, que le quedan al jugador, en el caso de que sea un objeto que tenga un desgaste.
Nuevo parámetro '**Usos**' en la entidad ItemObject
Nuevo comando QINVUSO

Usando el comando PINV ahora se puede indicar el nombre del objeto y la cantidad de usos (desgastes) que tiene.

Nuevo indicador que muestra el dinero del jugador.
Tiene las mismas propiedades que la puntuación: se gestiona mediante variables, se guarda al auto-salvar partida, etc...
Nueva variable del sistema DINERO
Nuevo parámetro '**Dinero**' en la entidad DeathMatchStart

Se introduce un fichero de configuración para Entidad 3D llamado Entidad3D.ini (aunque podrá ser distinto para cada juego en particular), donde se podrán indicar diversos parámetros para el funcionamiento del programa, sobre todo para las primeras funciones que realiza el programa antes de proceder con la carga del nivel.

Se permiten hasta 99 imágenes en la presentación inicial del juego (imágenes '_intro'). También es posible establecer un retardo de tiempo automático entre imágenes.
Nuevo parámetro '**Retardo_intro**' en el fichero de configuración Entidad3D.ini

Nuevos colores para el tipo de letra pequeña.
Nuevo tipo de letra **4** tamaño 3.
Nuevo tipo de letra **5** tamaño 3.
Nuevo tipo de letra **6** tamaño 3.

En la entidad ItemObject' se añaden los siguientes parámetros que permiten definir una descripción de 4 líneas del objeto.
'**Descrip1**'
'**Descrip2**'
'**Descrip3**'
'**Descrip4**'

Nuevo panel de descripción de objeto que presenta una descripción ampliada de los objetos del inventario del jugador.

Nuevo **panel de objeto** que muestra datos de los objetos situados con entidades '**ItemObject**' antes de ser recogidos o comprados por el jugador.
Nuevo parámetro '**PanelObjeto**' en la entidad ItemObject

En la entidad ItemObject se añade el parámetro '**Boton_accion**' para condicionar la recogida (o compra) del objeto y el procesamiento del fichero de comandos .INI, a la pulsación de la tecla '**Acción**' por parte del jugador.

En la entidad ItemObject se añaden los siguientes parámetros que fijarán los precios de Compra y venta de objetos.
'**Precio_compra**'
'**Precio_venta**'

Nuevo comando QINVVENDER que permite realizar, de forma automática, toda la operación de venta de un objeto del inventario del jugador.

Posibilidad de realizar un teletransporte del jugador hacia cualquier posición del mapa fijada con una entidad Posicion.
Nuevo comando TELETRANSPORTE
Nuevo parámetro '**Num_posicion**' en entidad Posicion

Nueva opción que permite al motor 3D no calcular los modelos 3D situados con entidades 'SpecificActor' y que el jugador no tiene a la vista.
Esto permite aumentar ligeramente el rendimiento del programa en algunas ocasiones.
Nuevo parámetro '**RenderSiNoVisible**' en entidad SpecificActor

Armas de energía.
Es un nuevo tipo de arma que sale del jugador en todas direcciones y daña a los enemigos que alcanza.
El efecto visual de estas armas puede ser definido mediante imágenes BMP en secuencia para crear animaciones.
Estas armas se activan mediante comandos, por lo que se podrán activar usando cualquier entidad que soporte comandos, aunque principalmente están pensadas para usar pulsando teclas, botones del ratón o usando objetos del inventario del jugador.
Nueva entidad Fx_ArmaEnergia
Nuevo comando ARMAENERGIA

Nuevo parámetro '**InmuneEnergia**' en entidad BotActorStart

Resta salud al jugador en las caídas desde diversas alturas.
Nuevo parámetro '**QuitaSaludCaidas**' en entidad DeathMatchStart

Barras indicadoras configurables.
Hasta 10 barras simultáneamente en pantalla (horizontales o verticales).
Cada barra puede estar definida por su propio BMP de 256 colores o RGB (ambos con transparencia).
Controlables mediante comandos o automáticamente por su propio temporizador.
Lanza un fichero de comandos al llenarse por completo y otro distinto al vaciarse por completo.
Nueva entidad MarcadorBarra
Nuevos comandos para controlar las barras:
'BARRAACTIVAR'
'BARRADESACTIVAR'
'BARRAVISIBLE'
'BARRAINVISIBLE'
'BARRAVALOR'

Variación, durante el juego, de la altura a la que puede saltar el jugador.
Se puede establecer la nueva altura que se quiera y por cuanto tiempo podrá saltar a esa nueva altura (o por tiempo indefinido).
Nuevo comando SALTO

Se puede asociar una animación del personaje protagonista para cuando escale bloques marcados con 'ladder' en World Editor.
Nuevo parámetro '**Anim_escalar**' en entidad DeathMatchStart.

En la entidad ItemObject, nuevo parámetro 'Transparencia' que permite establecer el nivel de transparencia del objeto.

Posibilidad de establecer la velocidad a la que se desplaza el jugador.
También existe la posibilidad de variar la velocidad durante el transcurso del juego.
Nuevo parámetro '**Velocidad**' en entidad DeathMatchStart
Nuevo comando VELOCIDAD

Cuando el jugador avance agachado se desplazará a la mitad de velocidad que lo hace cuando está corriendo.

Nuevo comando SALIRJUEGO que permite salir del programa Entidad 3D limpiamente y sin presentar ningún tipo de mensaje.
Este comando permite definir un tiempo de retardo antes de salir y también existe la posibilidad de poder indicar una aplicación externa a lanzar justo después de salir del programa.
Esto permite controlar el lanzamiento de niveles de Entidad 3D desde una aplicación externa, como podría ser un programa .EXE realizado con aplicaciones como Macromedia Flash.

En el fichero de seguimiento de procesos informe_Entidad_3d.log, se añaden diversos datos sobre las características del PC.

Se incorporan diversas protecciones para que no se puedan hacer trampas en los juegos:
Posibilidad de encriptar los ficheros .INI y .DLG para la distribución final del juego, para que su contenido no pueda ser visto por el jugador.
Posibilidad de impedir que el jugador pueda hacer trampas en el juego usando los comandos de la consola.
Posibilidad de impedir que el jugador comience a jugar por otro nivel que no sea por el primero.
Protección contra la manipulación de lo ficheros .dat de partida salvada.
Protección contra la manipulación de los ficheros .bsp de los niveles.

Nuevos comandos MEM y STATS para utilizar en la consola de juego.

Para más información sobre estas nuevas características, consulta las secciones de los comandos y las entidades correspondientes que encontrarás en este mismo manual.

Nuevas características de Entidad 3D v3.4c

A continuación se describen las nuevas características que se han implementado en la versión 3.4c con respecto a la versión 3.4:

Carga correcta de partidas autosalvadas (hay que borrar previamente todos los ficheros .dat de la carpeta 'partidas').

Se resta menos salud al jugador en las caídas desde diferentes alturas.

Opción para que los enemigos mueran presentando un efecto de explosión.
Nuevo parámetro '**ExplosionMorir**' en entidad 'BotActorStart'

Liberar memoria basura entre niveles.
Opción que permite liberar memoria residual de Windows durante el juego.
Nueva opción '**Liberar_memoria_basura**' en el fichero de configuración de Entidad 3D 'entidad3d.ini'

Para más información sobre estas nuevas características, consulta las secciones de los comandos y las entidades correspondientes que encontrarás en este mismo manual.

Nuevas características de Entidad 3D v3.5a

A continuación se describen las nuevas características que se han implementado en la versión 3.5a con respecto a la versión 3.4c:

Se modifica el funcionamiento del comando 'SALIRJUEGO' para que pueda salvar la partida antes se salir.
Nuevo parámetro '**nivel**' en el comando 'SALIRJUEGO'
¡ATENCION!: Esta modificación afecta a juegos creados con Entidad 3D v3.4c o inferior que usen el comando 'SALIRJUEGO'. Te recomiendo modificar todos los comandos 'SALIRJUEGO' que hayas podido utilizar, adaptándolos a la nueva sintaxis de tres parámetros.

El jugador ahora tiene la posibilidad de volar.
Nueva entidad 'VolarJugador'
Nuevo comando 'ENERGIAVOLAR'

Nuevo truco activable desde la consola de juego que permite volar atravesando paredes, techos y suelos.
Nuevo comando de consola 'VOLAR'

Permite salvar variables en un fichero para poderlas recuperar posteriormente en cualquier momento y desde cualquier nivel.
Nuevos comandos 'SALVAVARIABLE' y 'CARGAVARIABLE'

Para más información sobre estas nuevas características, consulta las secciones de los comandos y las entidades correspondientes que encontrarás en este mismo manual.

Nuevas características de Entidad 3D v3.6a

A continuación se describen las nuevas características que se han implementado en la versión 3.6a con respecto a la versión 3.5a:

Soporte del parámetro -PrevMode en la línea de ejecución de Entidad 3D, que permite lanzar niveles usando el 'driver' de pantalla y resolución que se seleccionaron la última vez que se usó el parámetro **-PickMode** en la línea de ejecución de Entidad 3D.

Disponibles resoluciones de hasta 1680x1050 tanto en Direct X como en OpenGL.

Mejora en el sistema de puntería de los enemigos para las armas número 1 (Blaster) y número 3 (Lanza-Cohetes).
Antes, el jugador podía ocultarse tras una pared que le llegaba por la cintura y los disparos de los enemigos nunca le alcanzaban.

Nuevo parámetro '**model**' en la entidad Fx_Surtidor permite asociar esta entidad a 'models' del escenario para que el efecto pueda seguir el movimiento de bloques 'models'.

Mejoras en la entidad CamaraFija.
Nuevo parámetro '**model**' en la entidad '**CamaraFija**' permite asociar cámaras a 'models' del escenario para poder crear cámaras móviles que sigan la ruta que hayamos establecido para bloques 'models' del escenario.
Nuevo parámetro '**Apuntar**' en la entidad CamaraFija permite orientar la cámara hacia entidades 'Posicion' del escenario.

Nuevo comando JUGADORVOLAR1 para activar el modo volar del jugador.

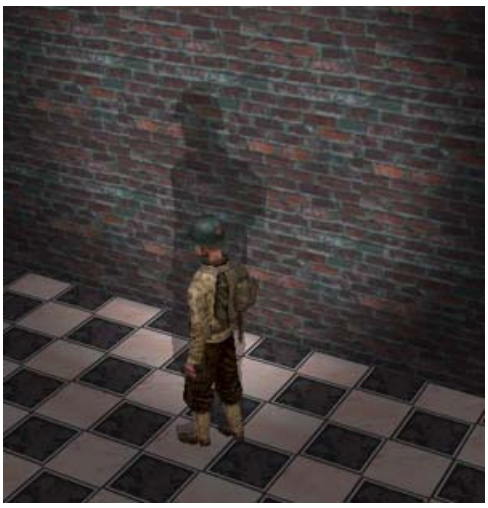
Nuevo comando JUGADORVOLAR0 para desactivar el modo volar del jugador.

Se pueden definir zonas críticas en los enemigos, donde un impacto directo de una arma del jugador podrá causar la muerte al instante del enemigo (cabeza, corazón, etc...).
En entidad BotActorStart, nuevos parámetros '**Min_zona_critica**' y '**Max_zona_critica**'.
En entidades ConfigArmaUno, ConfigArmaTres y ConfigArmaCuatro, nuevo parámetro '**Resta_salud_critica**'.

Soporte para luces dinámicas tipo foco.
En la entidad DynamicLight nuevos parámetros '**Orientacion**' y '**Arc**' para definir hacia donde orientar el foco de luz y para establecer el ángulo de apertura del mismo.
Hasta ahora sólo existían luces tipo foco (entidades '**spotlight**') de tipo estático, es decir, sus características no podían ser modificadas durante el transcurso del juego y la luz que desprendían no afectaba a bloques móviles ni a personajes durante el juego.

La luz que lleva el jugador, a parte de poder ser como hasta ahora de tipo antorcha (ilumina hacia todos los lados), ahora puede ser de tipo linterna, por lo que sólo iluminará hacia donde mire el jugador y de la manera que lo hace una linterna.
En entidad LuzJugador nuevo parámetro '**Arc**'.

Soporte de sombras dinámicas para modelos 3D en formato .ACT:
En entidades DeathMatchStart, ConfigEnemigos, SpecificActor, ItemHealth, ItemVida, ItemArmor y ItemObject nuevo parámetro '**Sombra**'.
En entidad Configuracion, nuevo parámetro '**SombrasDinMax**'.
Para una correcta visualización de este tipo de sombras, se recomienda siempre usar el 'driver' de pantalla '(D3D)Controlador de pantalla primaria' a **32bpp**.



La iluminación de la luces estáticas (entidades '**light**' y '**spotlight**') ahora también afectan a todos los modelos .ACT, no sólo a la hora de proyectar sombras dinámicas si no en el modo en que afectan los colores de las luces en el propio modelo .ACT.

Posibilidad de que el jugador no desprenda sangre por impactos de arma número 4.
En entidad DeathMatchStart nuevo parámetro '**Sangre**'.

Para más información sobre estas nuevas características, consulta las secciones de los comandos y las entidades correspondientes que encontrarás en este mismo manual.

Nuevas características de Entidad 3D v3.6b

A continuación se describen las nuevas características que se han implementado en la versión 3.6b con respecto a la versión 3.6a:

Se eliminan temblores de la cámara en tercera persona cuando el jugador usaba ascensores.

Si se usa el comando 'SALIRJUEGO' para saltar al siguiente nivel, ahora se conservan las vidas y el dinero que tenía el jugador en el nivel anterior. También funciona correctamente la opción 'Nueva Partida' que no cargaba el primer nivel del juego si se había usado el comando 'SALIRJUEGO'.

En la entidad 'Configuracion', nuevo parámetro '**Camara_RPG_alt**' que permite regular la altura de la cámara tipo RPG.

Posibilidad de procesar ficheros de comandos al detectar impactos de armas en el enemigo.
Nuevo parámetro '**Comandos_impacto**' en la entidad BotActorStart.
Nuevo parámetro '**Pierde_salud**' en la entidad BotActorStart.

Posibilidad de cambiar el personaje protagonista en cada nivel.
Basta con dejar en la carpeta '**actors**' el fichero .ACT del personaje con el nombre del ***nivel*** seguido del símbolo _ y a continuación **player1.act** o **player2.act**
Por ejemplo: Para cambiar el personaje protagonista en el nivel llamado '**caverna**', bastará con dejar el fichero .ACT del personaje con el nombre **caverna_player1.act**.

Corregido 'bug'. El jugador ahora se puede agachar y poner de pie correctamente en zonas de bloques vacíos 'empty'.

Se permite establecer número de vidas igual a 0 en la entidad 'DeathMatchStart', para que el jugador sólo disponga de una vida al comenzar el juego.

Corregido 'bug' que mostraba sombras circulares donde no debían aparecer.

Nuevo comando EXEC para ejecutar programas externos durante el transcurso del juego.

Corregidas las posiciones donde impactan los disparos y la posición de origen de los proyectiles de las armas, cuando el jugador dispara agachado.

Para más información sobre estas nuevas características, consulta las secciones de los comandos y las entidades correspondientes que encontrarás en este mismo manual.

Nuevas características de Entidad 3D v3.7a

A continuación se describen las nuevas características que se han implementado en la versión 3.7a con respecto a la versión 3.6b:

Posibilidad de generar niveles al azar.
Nuevas opciones en el mismo menú de selección de resolución de pantalla, que permiten generar mazmorras y laberintos 3D.

Posibilidad de lanzar ficheros de comandos al detectar impactos de disparos en 'models' del escenario.
Nuevo parámetro '**CommandsFile2**' en la entidad CommandModel.

Posibilidad de que el jugador pueda abrir puertas y activar elevadores o plataformas por impactos de sus disparos, pero bajo condiciones que podremos establecer mediante ficheros de comandos.
Nuevos parámetros '**CommandsFile2**' y '**AbrirPorImpacto**' en la entidad Door.
Nuevos parámetros '**CommandsFile2**' y '**MoverPorImpacto**' en la entidad MovingPlat.

Los enemigos pueden abrir puertas y usar elevadores o plataformas, pero bajo condiciones que podremos establecer mediante ficheros de comandos.
Nuevos parámetros '**CommandsFileE**' y '**EnemigoAbre**' en la entidad Door.
Nuevos parámetros '**CommandsFileE**' y '**EnemigoMueve**' en la entidad MovingPlat.

Posibilidad de presentar en pantalla el número de vidas de las que dispone el jugador en forma de representación numérica.
Nuevo parámetro '**VidasNum_XY**' en la entidad Marcador.

Se añaden las siguientes utilidades en la presente distribución de Entidad 3D: **World Editor Pro**, **Dungeon Maker**, **Entidad 3D Actor Viewer** y **Skin replacer**.

Para más información sobre estas nuevas características, consulta las secciones de los comandos y las entidades correspondientes que encontrarás en este mismo manual.

Nuevas características de Entidad 3D v3.7d

A continuación se describen las nuevas características que se han implementado en la versión **3.7d** con respecto a la versión **3.7a**:

Corrige 'bug' que no permitía eliminar el charco de sangre del jugador muerto.

Corrige 'bug' que en ocasiones podría ser el causante de salidas inesperadas del programa cuando los enemigos o cualquier modelo .ACT son alcanzados por impactos de armas.

Corrige 'bug' que si el jugador moría estando agachado, en la siguiente vida la velocidad a la que corría aún era la velocidad a la que se corre agachado.

En lugar de que el jugador use pulsar disparar para utilizar la siguiente vida cuando ha muerto, ahora se debe pulsar saltar. De esta forma se evita un disparo que en muchas ocasiones se producía al usar la siguiente vida.

Los enemigos y el jugador tienen ahora una animación de muerte diferente cada vez que mueren (de las 4 disponibles).

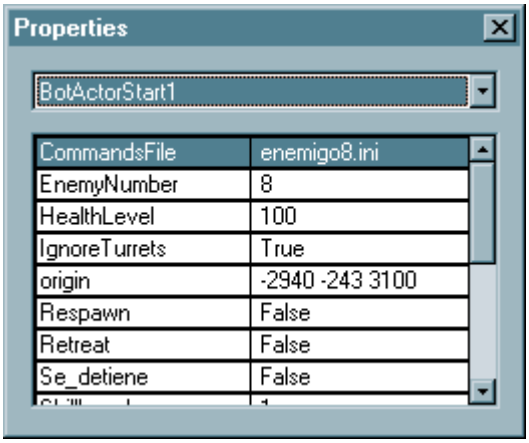
Los comandos **PERS CHKP** y **DVID** funcionan usando la entidad **TeclaComandos**.

Para más información sobre estas nuevas características, consulta las secciones de los comandos y las entidades correspondientes que encontrarás en este mismo manual.

Utilización de niveles de anteriores versiones de Entidad 3D

Si quieres que por lo menos te funcionen los niveles que habías diseñado con anteriores versiones de Entidad 3D tendrás que realizar unos sencillos pasos para poder conseguirlo.
Los siguientes pasos no son para dotar a tus antiguos niveles con las nuevas características de la nueva versión de Entidad 3D, sino para que tus niveles funcionen en el nuevo entorno de Entidad 3D por lo menos como lo hacían en el antiguo entorno:

1.- Ejecuta el programa World Editor que tengas instalado para la versión superior de Entidad 3D.
2.- Desde este World Editor carga tu antiguo nivel.



3.- Entra en la lista de entidades de tu nivel pulsando y selecciona la primera entidad de la lista. Seguidamente, mediante la tecla 'abajo', sin salir de la ventana 'Properties', ves seleccionando el resto de entidades hasta la última:

De lo que se trata es simplemente de ir viendo todas las entidades de tu antiguo nivel, al hacer esto World Editor las va actualizando con los nuevos parámetros de la nueva versión.
4.- Finalmente compila tu nivel y sálvalo.

Si has actualizado tu versión de Entidad 3D, recuerda que también deberás hacer lo mismo con los niveles de ejemplo que pudiesen existir de la anterior versión si quieres que te funcionen con la nueva versión.

Preguntas y respuestas más frecuentes

Pregunta: Tengo una tarjeta gráfica Voodoo, por lo tanto con chips 3DFX aceleradores 3D y me gustaría utilizar los drivers 'Glide' nativos de esta tarjeta para acelerar al máximo el rendimiento de Entidad 3D.

Respuesta: En la carpeta llamada 'DLL' que parte de la carpeta principal de Entidad 3D (por defecto 'entidad_3d') encontrarás el fichero 'glidedrv.dll', cópialo en la carpeta principal donde se encuentra 'Entidad3D.exe'. La próxima vez que arranques un escenario ya se te activará automáticamente el modo nativo de tu tarjeta gráfica.

Pregunta: ¿ Necesita un usuario tener instalado en su PC algún tipo de 'run-time' o el mismo Entidad 3D para sólo jugar a un juego creado con Entidad 3D ?.

Respuesta: El jugador no necesita nada más que el conjunto de ficheros que le suministraremos en un archivo .ZIP. En realidad ahí va parte de Entidad 3D, pero el usuario de nuestro juego no necesita saber con que se ha creado el juego. Sólo tendremos que decirle que fichero .BAT ha de 'clickar' para dar comienzo el juego. Tampoco estaría de más que le informásemos de los controles de teclado y ratón que usa el juego.

Lo que si es necesario es que tenga instalado en su PC los 'drivers' 'Microsoft Direct X 8.1' o superior.

Pregunta: Al cargar un nivel y justo después de ver en pantalla 'Cargando nivel, esperar ...', el programa sale repentinamente a Windows sin presentar ningún tipo de error.

Respuesta: Normalmente esto puede ocurrir por dos motivos:
1.- En 'World Editor' tienes una entidad 'Door' o 'MovingPlat' con el parámetro 'Model' cargado con 'null' o indicando un modelo que no existe porque lo has borrado.
2.- En 'World Editor' has utilizado una textura que no cumple con las dimensiones: 64x64, 128x128 o 256x256.

Pregunta: Al cargar un nivel y justo después de ver en pantalla 'Cargando nivel, esperar ...', el programa sale y me presenta el error: BlockActor_Spawn: No model for entity:BlockActor.

Respuesta: En tu nivel, una o más entidades 'BlockActor' las has dejado con el parámetro 'Model' como 'null'.
Revisa las entidades que tienes en tu nivel y no dejes ninguna con 'Model' a 'null'.

Pregunta: ¿ Puedo añadir archivos de video (.mov, .avi, etc) para una intro de un juego, final o momento importante ?

Respuesta: No se puede

Pregunta: ¿ Qué significan los mensaje 'Face with no normal' que me aparecen en la consola de 'World Editor' cuando realizo la compilación de mi nivel ?

Respuesta: Son avisos de que algún brush no está bien encajado, de todas formas son avisos inofensivos para el buen funcionamiento del escenario.

Pregunta: Quiero hacer un juego donde no se utilicen armas y me gustaría poder desconectar el arma Blaster que siempre aparece por defecto en las manos del jugador al comenzar el juego.

Respuesta: Utiliza la entidad 'Configuracion', existe un parámetro en esa entidad que te permitirá realizarlo.

Pregunta: ¿ Se pueden usar armas de cuerpo a cuerpo ?

Respuesta: Si, utilizando la entidad '**ConfigArmaCuatro**'.

Pregunta: ¿ Es posible poner el valor de rotación del sky en 0, es decir que el cielo esté fijo.

Respuesta: Claro, así está puesto en el juego de ejemplo 'Secta'. Asegúrate de pulsar 'Enter' en la caja cuando hayas puesto 0 en la rotación.

Pregunta: En World Editor. ¿ Por qué al marcar como AREA a una puerta, al compilar me aparece el mensaje 'WARNING* FinishAreas: AreaPortal only touched one area' ?.

Respuesta: El problema es que la puerta no tapa por completo el hueco que ocupa por lo que el motor 3D no podrá trabajar eficientemente esa 'Area' que has marcado para que no la calcule cuando la puerta está cerrada, ya que efectivamente estará cerrada pero por alguna abertura se podrá ver el otro lado.

Pregunta: ¿ Cual es la forma correcta de utilizar la entidad "changelevel" que es para saltar a otro nivel, ya que la utilizo y no me funciona ?.

Respuesta: El 'model' que crees para el 'changelevel' hazlo que sea algo como una plataforma muy baja a la que pueda subirse el jugador, de esta manera te funcionará si el jugador se sube a la plataforma.

Pregunta: ¿ Se pueden hacer sonidos en 3D posicionales en mi escenario ?

Respuesta: Si, consulta en el manual de Entidad 3D, no en el manual de World Editor, el modo de funcionamiento de la entidad 'ElectricBolt'.

Pregunta: Cuando arranco un juego, el juego de ejemplo 'Secta' o uno mío, incluso en bajas resoluciones, anda lento. Tengo una placa de video Nvidia Geforce MX 4 440 de 64mb con los controladores actualizados.

Respuesta: Entidad 3D no sólo tira de tarjeta 3D, sino también de CPU. De todas formas comprueba las opciones que tengas activadas o desactivadas en tu configuración de Direct 3D.
Prueba desactivar la opción de 'vsync' en la configuración de tu placa de video.

Pregunta: Cuando coloco enemigos en mi nivel (con la entidad 'BotActorStart') ellos no me disparan y se quedan quietos, ni siquiera me persiguen.

Respuesta: La cruz que representa esta entidad en 'World Editor' es la posición de los pies del enemigo, por lo que intenta colocarla cerca del suelo, pero sin tocarlo. Si lo pones demasiado cerca del suelo verás que el enemigo ni se mueve ni dispara al jugador.

Pregunta: ¿ Entidad 3D incluye soporte para jugar vía red TCP/IP, Netbios o puerto serie ?

Respuesta: No, no tiene disponible ninguna opción de juego on-line.

Pregunta: ¿ Es posible crear personajes que cooperen con el jugador para realizar misiones o que pudieran ayudar a matar enemigos ?

Respuesta: No es posible.

Pregunta: En el ascensor que tengo en mi mundo, me pongo debajo y cuando baja me atraviesa y no me aplasta quitándome salud.

Respuesta: En la parte de abajo del ascensor agrégale un brush delgado con 'lava'.

Pregunta: Intento arrancar el juego de ejemplo 'Secta' y me aparece este mensaje 'GameMgr_SetWorld: geWorld_Create falló: Levels\secta.BSP'. También me pasa en mis niveles si son muy grandes.

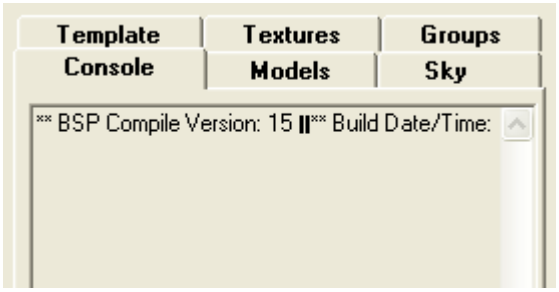
Respuesta: Entidad 3D no dispone de la memoria suficiente para cargar los niveles. Intenta liberar memoria de tu sistema.

Pregunta: Quiero diseñar mis propios personajes y objetos 3D, para luego utilizarlos en Entidad 3D. ¿ Qué programa he de utilizar ?.

Respuesta: 'MilkShape 3D' es el mejor programa que se adapta a las necesidades de Entidad 3D, ya que soporta los formatos de ficheros utilizados por Entidad 3D y World Editor.
Milkshape 3D es un programa Shareware, pero os aseguro que vale de sobras los 25 euros que cuesta registrarse. Si utilizas una versión no registrada sólo te permitirá salvar tus proyectos durante 30 días.

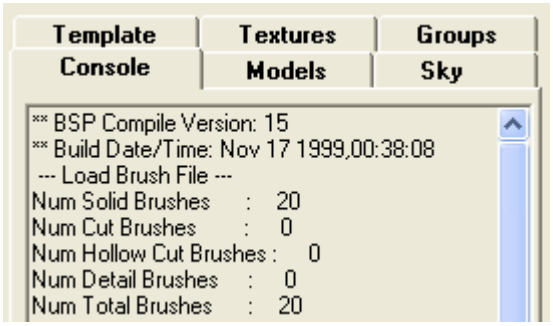
Pregunta: En Windows 2000 y XP al recompilar el nivel de ejemplo 'secta', en la consola de World Editor me aparece este error:
GPreviewPath: c:\entidad_3d\entidad3d.exe
CopyFile (.\secta.bsp, c:\entidad_3d\levels\secta.bsp)
GetLastError()==32

Respuesta: Al compilar el nivel en En World Editor, cuando te aparezca la ventana 'Compile Manager', en la caja 'MAP File:' cambia la ruta '.\secta.map' por la ruta completa 'c:\entidad_3d\levels\secta.map'



Pregunta: En Windows XP la consola de World Editor no me muestra la evolución de la compilación, me aparece siempre esto:

Respuesta: Para el programa 'gedit.exe' has establecido compatibilidad para Windows 98. Vuelve a dejarlo como estaba, sin utilizar la opción de compatibilidad y te funcionará perfectamente:



Pregunta: En World Editor, al compilar siempre me aparece este error en la consola:
Compile failed: GBSP_CreateBSP encountered an error, GBSPLib.DLL.

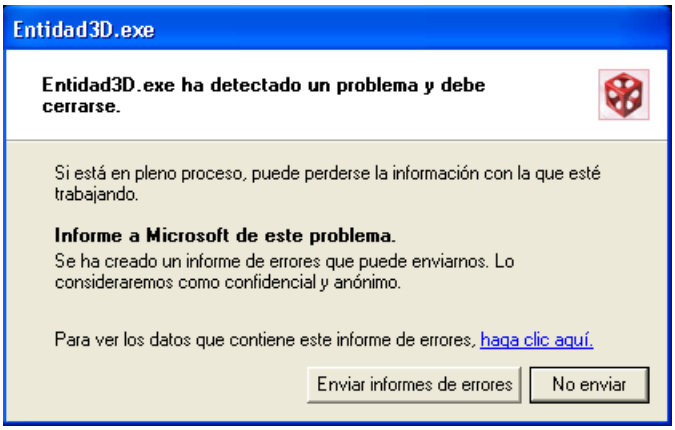
Respuesta: Debes añadir al menos la entidad 'DeathMatchStart' a tu nivel.

Pregunta: En World Editor me dispongo a crear un nivel pero no me aparecen muchas entidades en la lista de entidades disponibles, entre ellas 'DeathMatchStart', 'Door' 'Command', etc...:

Respuesta: Para comenzar un nuevo nivel debes cargar antes el nivel llamado 'vacio', entonces te aparecerán todas las entidades disponibles.
Si te ocurre con niveles antiguos sólo tienes que cargarlos y en 'Options/Level Options' dejar 'Head Directory' como esto:
.\src_entidad3d;.\src_entidad3d\fx;.\src_entidad3d\game
y 'Texture Library' con tu fichero de texturas, por defecto como esto:
.\texturas.txl

Pregunta: Al ejecutar "Entidad3D.exe" me aparece un error que se reporta en el archivo "Ent3d.Log" y el contenido de ese archivo es:
Error#: 0, Code#: 0, Info: GameMgr_SetWorld: geVFile_Open falló. No se puede abrir el fichero: .
Respuesta: Estas ejecutando 'entidad3d.exe' directamente. Esto no lo debes hacer a menos que sepas como usar el fichero 'autonivel.ini'.
Las instrucciones de instalación dicen que una vez instalado Entidad 3D utilices World Editor (gedit.exe) para comenzar a crear tus escenarios. Te recomiendo que sigas el tutorial de World Editor paso a paso.

Pregunta: Al dar comienzo mi nivel llamado 'bosque', justo después del logotipo de Genesis3D, me aparece este error:
Error#: 2, Code#: 0, Info: GameMgr_SetWorld: geWorld_Create falló: Levels\bosque.BSP.
Error#: 1, Code#: 0, Info: Entities.c 958
Error#: 0, Code#: 0, Info: Entities.c 864
Respuesta: Esto lo provoca una entidad que estás utilizando en tu nivel que utiliza el parámetro 'Model'. El problema está en el 'model' al que te estás refiriendo en esa entidad. Lo más probable es que te estés refiriendo a un 'model' que no existe, o el 'model' carece de animación, o el 'model' no está bien asociado a 'brushes', o el 'model' lo tienes a 'null' mientras tendría que contener un 'model' válido.
Básicamente te recomiendo que te revises el 'model' implicado en la entidad que lo utiliza.



Pregunta: Intentando iniciar cualquier nivel, incluidos los de ejemplo, me aparece este error justo después del mensaje 'Cargando...':

Respuesta: Es un error muy genérico de Windows XP, pero una de las causas de este error en Entidad 3D es que estás intentando cargar un nivel que se desarrolló con otra versión de Entidad 3D. Seguramente has actualizado Entidad 3D y no has recompilado los niveles siguiendo los pasos que se recomiendan en la sección '**Actualizar Entidad 3D**' en la misma página de Entidad 3D o en la sección 'Utilización de niveles de anteriores versiones de Entidad 3D' de este mismo manual.

Pregunta: En 'World Editor' al compilar mi nivel me aparece este error en la consola y el nivel no se llega a compilar:
PrepAllGBSPModels: Could not prep model 0.
ConvertGBSPToFile: Could not Prep Models.
Respuesta: Estás marcando algún 'brush' con 'Area' cuando en realidad ese 'brush' no separa dos áreas del nivel. Comprueba en el manual de 'World Editor' para que sirve marcar 'brushes' con 'Area'.

Pregunta: Cuando compilo con 'World Editor', aparece un error de compilacion y en la consola el siguiente mensaje:
ConvertGBSPToFile: geVFile_OpenNewSystem failed.
Compile Failed: GBSP_SaveGBSPFile for file: C:\Archivos de programa\Entidad\levels\rampas.bsp, GBSPLib.Dll.
Respuesta: Instala Entidad 3D en la carpeta que se recomienda (**c:\entidad_3d**) y no en 'Archivos de programa ...', a World Editor no le gustan los nombres o carpetas con espacios.

Pregunta: Al intentar abrir el fichero de nivel '**c:\entidad_3d_3.5a\levels\minivel.3dt**' con 'World Editor', me aparece el mensaje de error **Expected 'NumPoints' in file ...**, y no abre el fichero:
Respuesta: Instala Entidad 3D en la carpeta que se recomienda (**c:\entidad_3d**) y no en 'c:\Archivos de programa ...' o 'c:\entidad_3d_3.5a'. A World Editor no le gustan los nombres o carpetas con espacios o puntos.

Pregunta: Al intentar abrir un fichero de nivel .3DT con 'World Editor', me aparece el mensaje de error **Expected 'NumEntities' in file ...**, y no abre el fichero:
Respuesta: Estás intentando abrir un fichero .3DT con una versión de World Editor anterior a la que se utilizó para crear ese fichero de nivel .3DT

Licencias:

Licencia de uso de Entidad 3D (Freeware)

Licencia de uso del juego 'Secta' (Freeware)

Licencia comercial

Genesis3D SDK

Librería Bass

Librería FreeImage

ENTIDAD 3D v3.7 - LICENCIA Y CONDICIONES DE USO PARA LA VERSION 3.7 DEL PROGRAMA ENTIDAD 3D:

INFORMACION AL USUARIO: Lea atentamente el siguiente acuerdo de Licencia de Uso. La utilización del software proporcionado con este acuerdo supone que Usted acepta los términos del mismo. Si Usted no está de acuerdo con los términos de este acuerdo, elimine inmediatamente de su ordenador, este software junto con todas sus copias. El uso de este software, sus utilidades o su documentación conlleva la aceptación de todos estos términos y condiciones.

La presente versión de Entidad 3D es FREEWARE, así como toda la documentación que se incluye. No ocurre lo mismo con las utilidades que acompañan a la presente distribución de Entidad 3D, llamadas World Editor, Texture Packer, Actor Studio, Actor Viewer, Entidad 3D Actor Viewer, Skin replacer y Dungeon Maker, siendo estos programas 'OpenSource' o Freeware y rigiéndose según las licencias particulares de cada una de estas utilidades. Varias de estas utilidades son propiedad de Eclipse Entertainment aunque se distribuyen junto a Entidad 3D para facilitar el desarrollo de juegos con Entidad 3D.
FREEWARE quiere decir que Usted puede instalar y utilizar este software en su PC por un periodo de tiempo ilimitado para usos no comerciales. Debido a la naturaleza freeware de este software, éste se proporciona sin coste alguno. Freeware no es lo mismo que software de dominio público, el autor del programa, en este caso 'Jordi Pérez Barreiro', mantiene el copyright sobre el software y puede establecer restricciones sobre la utilización del mismo.

1.- CONCESION DE LICENCIA. De acuerdo con los términos de este acuerdo, el autor de este software le otorga a Usted una licencia no exclusiva que le permitirá usar y reproducir el programa sin restricciones, siempre y cuando todas y cada una de las copias realizadas contengan todos los avisos de propiedad sin modificaciones, incluyendo, pero sin estar limitado al mismo, este acuerdo de licencia. Esta licencia no incluye el derecho a recibir ningún soporte técnico, actualizaciones o cualquier otro servicio.

2.- PROPIEDAD INTELECTUAL. Tanto el programa, como los correspondientes copyrights u otros derechos de propiedad industrial o intelectual, son propiedad del autor del programa y están protegidos por las leyes españolas de Propiedad Intelectual (ley 22/1987 de 11 de Noviembre sobre Propiedad Intelectual y el artículo 270 del Código Penal) y por las disposiciones de los tratados internacionales que sean de aplicación.

3.- RESTRICCIONES. Usted reconoce y acuerda que no podrá:
(a) intentar desactivar el software de cualquier modo o manera;
(b) intentar descompilar, desensamblar, realizar ingeniería inversa o intentar derivar el código fuente del programa;
(c) modificar código o datos del programa que están contenidos en el fichero 'entidad3d.exe';
(d) vender, arrendar, alquilar o revelar ninguna parte del programa a un tercero;
(e) retirar o alterar cualquier marca registrada, logotipo, copyright o cualquier otro aviso; leyenda, símbolo o etiqueta de propiedad contenida en el programa o producto;

4.- REDISTRIBUCION COMPLETA DEL PROGRAMA. El autor de este software le autoriza a Usted la redistribución completa del programa bajo las siguientes condiciones:
(a) que sea el fichero original sin modificaciones de ningún tipo;
(b) en Internet, que sea accesible al usuario sin que sea necesario registrarse en la 'web' que ofrece el programa ni que requiera ningún tipo de condición para cualquier usuario que desee descargarse el programa;
(c) que el programa se ofrezca de forma completamente gratuita;
(d) que se cite el autor del programa así como la dirección en Internet de la 'web' oficial del mismo.

El autor del programa autoriza la redistribución del mismo en CD, DVD u otros soportes multimedia que acompañen a revistas, ya sean de pago o no.

5.- REDISTRIBUCION PARCIAL DEL PROGRAMA. Debido a las características de Entidad 3D, cuya finalidad es crear juegos que se puedan distribuir libremente, el autor del programa autoriza la distribución de todos los ficheros que sean necesarios para el correcto funcionamiento del nuevo juego creado, bajo las siguientes condiciones:
(a) que la distribución no contenga los juegos de ejemplo de Entidad 3D;
(b) que la distribución no contenga ni la documentación ni los tutoriales de Entidad 3D;
(c) que la distribución no se anuncie que es el programa Entidad 3D;
(d) que el nuevo juego se ofrezca de forma completamente gratuita, a menos que el autor del juego disponga de una licencia de uso comercial de Entidad 3D 3.7, en cuyo caso queda a elección del autor del nuevo juego vender, arrendar o alquilar el nuevo juego creado a un tercero.

En cualquier caso queda a elección del autor del nuevo juego citar o no a Entidad 3D como la herramienta utilizada en la creación del nuevo juego.

6.- GARANTIA LIMITADA. El autor del programa no ofrece ninguna garantía con respecto al mismo. El programa se distribuye tal como es, no aceptándose reclamaciones por supuestas especificaciones que debiera cumplir el programa.

7.- AUSENCIA DE RESPONSABILIDAD POR DAÑOS INDIRECTOS. En ningún caso, el autor del programa será responsable por cualquier perjuicio supuestamente provocado por el uso, falta de uso o imposibilidad de uso del programa.

8.- LEY APLICABLE. El presente acuerdo se regirá conforme con las leyes de España, en caso de que surjan dudas y/o divergencias en cuanto a su interpretación y/o efectos, únicamente serán competentes los Tribunales de Barcelona, renunciando las partes a cualquier otro fuero que pudiera corresponderles. Si alguna cláusula de este acuerdo fuera contra la ley, será considerada como nula sin que ello afecte o suponga la nulidad de la totalidad del acuerdo.

9.- INDEMNIDAD. Usted accede a eximir y liberar de toda responsabilidad al autor del programa , sus sucesores, filiales, cargos, directores, agentes y empleados frente a cualquier reclamación o demanda -incluyendo, dentro de unos límites razonables, los gastos correspondientes a honorarios legales- planteada por una tercera parte, y que esté debida a o causada por un no cumplimento por su parte de los términos de este acuerdo, o por la violación por su parte de cualquier ley o derechos concernientes a una tercera parte.

SECTA - LICENCIA FREEWARE
CONDICIONES DE USO DEL PROGRAMA 'SECTA':

INFORMACION AL USUARIO: Lea atentamente el siguiente acuerdo de Licencia de Uso. La utilización del software proporcionado con este acuerdo supone que Usted acepta los términos del mismo. Si Usted no está de acuerdo con los términos de este acuerdo, elimine inmediatamente de su ordenador, este software junto con todas sus copias. El uso de este software o su documentación conlleva la aceptación de todos estos términos y condiciones.

El juego 'Secta', así como toda su documentación son FREEWARE. Esto quiere decir que Usted puede instalar y utilizar este software en su PC por un periodo de tiempo ilimitado. Debido a la naturaleza freeware de este software, éste se proporciona sin coste alguno. Freeware no es lo mismo que software de dominio público, el autor del programa, en este caso 'Jordi Pérez Barreiro', mantiene el copyright sobre el software y puede establecer restricciones sobre la utilización del programa.

1.- CONCESION DE LICENCIA. De acuerdo con los términos de este acuerdo, el autor de este software le otorga a Usted una licencia no exclusiva que le permitirá usar y reproducir el programa para uso personal o interno, siempre y cuando todas y cada una de las copias realizadas contengan todos los avisos de propiedad sin modificaciones, incluyendo, pero sin estar limitado al mismo, este acuerdo de licencia. Esta licencia no incluye el derecho a recibir ningún soporte técnico, actualizaciones o cualquier otro servicio.

2.- PROPIEDAD INTELECTUAL. Tanto el programa, como los correspondientes copyrights u otros derechos de propiedad industrial o intelectual, son propiedad del autor del programa y están protegidos por las leyes españolas de Propiedad Intelectual (ley 22/1987 del 11 de Noviembre sobre Propiedad Intelectual) y por las disposiciones de los tratados internacionales que sean de aplicación.

3.- REDISTRIBUCION DEL PROGRAMA. El autor de este software le autoriza a Usted la redistribución del programa bajo las siguientes condiciones:
(a) que sea el fichero .ZIP original sin modificaciones de ningún tipo;
(b) en Internet, que sea accesible al usuario sin que sea necesario registrarse en la 'web' que ofrece el programa ni que requiera ningún tipo de condición para cualquier usuario que desee descargarse el programa;

(c) que el programa se ofrezca de forma completamente gratuita;
(d) que se cite el autor del programa así como la dirección en Internet de la 'web' oficial del mismo.
El autor del programa autoriza la redistribución del mismo en CD, DVD u otros soportes multimedia que acompañen a revistas, ya sean de pago o no.

4.- RESTRICCIONES. Usted reconoce y acuerda que no podrá:
(a) redistribuir el programa en Internet bajo condición de que los usuarios tengan que registrarse en la página 'web' que ofrece el programa como premisa para descargar el programa;
(b) intentar desactivar el software de cualquier modo o manera;
(c) intentar descompilar, desensamblar, realizar ingeniería inversa o intentar derivar el código fuente del programa;
(d) vender, arrendar, alquilar o revelar ninguna parte del programa a un tercero;
(e) retirar o alterar cualquier marca registrada, logotipo, copyright o cualquier otro aviso; leyenda, símbolo o etiqueta de propiedad contenida en el programa o producto.

5.- GARANTIA LIMITADA. Debido al carácter freeware de este software, el autor del programa no ofrece ninguna garantía con respecto al mismo. El programa se distribuye tal como es, no aceptándose ninguna reclamación por supuestas especificaciones que debiera cumplir el programa.

6.- AUSENCIA DE RESPONSABILIDAD POR DAÑOS INDIRECTOS. El autor del programa no será responsable en ningún caso de los perjuicios supuestamente provocados por el uso o imposibilidad de uso del programa.

7.- LEY APLICABLE. El presente acuerdo se regirá conforme con las leyes vigentes en España. En caso de que surjan dudas y/o divergencias en cuanto a su interpretación y/o efectos, únicamente serán competentes los Tribunales de Barcelona, renunciando las partes a cualquier otro fuero que pudiera corresponderles. Si alguna cláusula de este acuerdo fuera contra la ley, será considerada como nula sin que ello afecte o suponga la nulidad de la totalidad del acuerdo.

8.- INDEMNIDAD. Usted accede a eximir y liberar de toda responsabilidad al autor del programa , sus sucesores, filiales, cargos, directores, agentes y empleados frente a cualquier reclamación o demanda -incluyendo, dentro de unos límites razonables, los gastos correspondientes a honorarios legales- planteada por una tercera parte, y que esté debida a o causada por un no cumplimento por su parte de los términos de este acuerdo, o por la violación por su parte de cualquier ley o derechos concernientes a una tercera parte.

Licencia comercial de Entidad 3D

Como ya habrás podido leer en la licencia de uso de Entidad 3D, este programa es completamente gratuito para usos no comerciales. ¿ Que quiere decir esto ?. Pues que puedes descargártelo de la 'web' de Entidad 3D y usarlo por un tiempo indefinido y sin restricciones para crear tus propios juegos, con la posibilidad de que los juegos que realices los puedas distribuir libremente, siempre y cuando no obtengas un beneficio económico con ello o lo utilices con fines comerciales.

Pero en el caso de que quieras utilizar Entidad 3D precisamente para realizar juegos que luego puedas vender o para obtener algún beneficio comercial, existe la posibilidad de adquirir una licencia comercial de Entidad 3D.

Para conseguir esta licencia tendrás que adquirirla pagando sólo 55 euros. Para realizar este pago ponte en contacto conmigo (Jordi Pérez) para informarte de las distintas modalidades de pago.

¿ Que obtienes adquiriendo una licencia comercial ?. Pues simplemente el derecho legal de vender tus juegos creados con Entidad 3D. En ningún caso, al comprar esta licencia, obtendrás una versión comercial de Entidad 3D, porque tal versión no existe, la versión de Entidad 3D que necesitarás para crear tus juegos comerciales será la misma que te has bajado de la 'web' de Entidad 3D, al ser esta ya una versión completa.

La licencia comercial tampoco te permitirá suprimir el logotipo inicial de Genesis3D que aparece cada vez que ejecutas un juego creado con Entidad 3D, la eliminación de este logotipo no depende de mí como autor de Entidad 3D sino de los creadores del motor 3D que usa Entidad 3D llamado Genesis3D.
Según se indica en la licencia de Genesis3D, este logotipo es posible eliminarlo si se paga cierta cantidad de dinero a los autores de Genesis3D, pero eso es un tema que tendrías que tratar directamente con ellos.

De todas formas, si posees una licencia comercial de Entidad 3D pero no de Genesis3D, podrás vender tus juegos sin problemas legales, yo lo garantizo con mi licencia y el propio Genesis3D también lo garantiza con la suya, siempre y cuando mantengas ese logotipo inicial de Genesis3D.

La licencia comercial que compres sólo es válida para la versión que en ese momento esté disponible de Entidad 3D, aunque tu licencia seguirá siendo válida para las siguientes actualizaciones hasta que Entidad 3D cambie de versión. Por ejemplo, si compras la licencia comercial de Entidad 3D v3.2 tendrás la licencia de esa versión y de las siguientes actualizaciones hasta la v3.9, por lo que podrás hacer juegos con esas actualizaciones y venderlos.
Las siguientes versiones tendrán un precio para la licencia comercial diferente, aunque si ya has comprado una licencia anteriormente sólo tendrás que abonar la diferencia del nuevo precio de la licencia comercial para seguir vendiendo tus juegos si los desarrollas con esa nueva versión de Entidad 3D

GENESIS3D PUBLIC LICENSE

Version 1.01

PREAMBLE

Please read this license carefully before using Genesis3D. By downloading and using Genesis3D, you are agreeing to be bound by the terms of this license. If you do not or cannot agree to the terms of this License, do not download or use Genesis3D.

This license is intended to make the source code of a state-of-the-art 3D engine available to everyone who wants the opportunity to use and contribute to a 3D engine. Genesis3D is a commercial product. It is not in the public domain or "freeware" or "shareware". By making the source code available we are not giving it away or abandoning our right of ownership. By using Genesis3D you agree to be bound by the spirit and the terms of this license.

This license permits you to use the Genesis3D SDK to create and sell products. You may also redistribute the Genesis3D SDK. You may also make modifications to the SDK and distribute them too.

If you modify the Genesis3D engine source, excerpt any portion of it, or if you use a modified version of the engine source, you must make your source code available for others to use under the same terms that we have granted you. This guarantees that modifications to the engine will be available to everyone and the best modifications can become part of the official Genesis3D release. Your source code must be released concurrent with releasing any executable built with Genesis3D. If you do not wish to make your source code available you must obtain a separate license from Eclipse.

Gtest, the sample game, is specifically excluded from this requirement. Feel free to modify Gtest and use it in any way that benefits you.

If you use the original unmodified Eclipse-built binaries there is no requirement for you to make your source code available.

You must display the original unmodified Genesis3D animated logo as the first logo on startup of your product, demo or application. You must also prominently display the Genesis3D logo on any marketing materials, advertising or packaging of your product.

These terms are good for the community because it makes the source of a high-end 3D engine available to everyone and it encourages sharing of improvements. They are good for Eclipse because some companies will opt to not share their source and will prefer to pay to license the engine instead. These kinds of licenses will fund future Genesis3D development and subsidize the free licenses that many people will enjoy.

1. DEFINITIONS

1.1. ``Contributor'' means each entity that creates or contributes to the creation of Modifications.

1.2. ``Contributor Version'' means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.

1.3. ``Covered Code'' means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof, and any Source Code which duplicates, enhances, modifies or replaces functionality of the Covered Code as defined in Exhibit B.

1.4. ``Electronic Distribution Mechanism'' means a mechanism generally accepted in the software development community for the electronic transfer of data.

1.5. ``Eclipse'' means Eclipse Entertainment, L.L.C.

1.6. ``Executable'' means Covered Code in any form other than Source Code.

1.7. ``Larger Work'' means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.

1.8. ``License'' means this document.

1.9. ``Modifications'' means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:
A. Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.
B. Any new file that contains any part of the Original Code or previous Modifications.

1.10. ``Original Code'' means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.

1.11. ``Source Code'' means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated interface definition files, scripts used to control compilation and installation of an Executable, or a list of source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor's choice. The Source Code can be in a compressed or archival form, provided the appropriate decompression or de-archiving software is widely available for no charge.

1.12. ``You'' means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 7.1. For legal entities, ``You'' includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, ``control'' means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of fifty percent (50%) or more of the outstanding shares or beneficial ownership of such entity.

2. SOURCE CODE LICENSE

2.1. The Eclipse Grant.
Eclipse hereby grants You a world-wide, fully paid, royalty-free, non-exclusive license, subject to third party intellectual property claims:
(a) to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, or as part of a Larger Work; and
(b) under patents now or hereafter owned or controlled by Eclipse, to make, have made, use and sell (``Utilize'') the Original Code (or portions thereof), but solely to the extent that any such patent is reasonably necessary to enable You to Utilize the Original Code (or portions thereof) and not to any greater extent that may be necessary to Utilize further Modifications or combinations.

2.2. Contributor Grant.
Each Contributor hereby grants You a world-wide, fully paid, royalty-free, non-exclusive license, subject to third party intellectual property claims:
(a) to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code or as part of a Larger Work; and
(b) under patents now or hereafter owned or controlled by Contributor, to Utilize the Contributor Version (or portions thereof), but solely to the extent that any such patent is reasonably necessary to enable You to Utilize the Contributor Version (or portions thereof), and not to any greater extent that may be necessary to Utilize further Modifications or combinations.

2.3. Your Grant.
You hereby grant to Eclipse a world-wide, fully paid, royalty-free, non-exclusive license, subject to third party intellectual property claims:
(a) to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by You (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code or as part of a Larger Work; and
(b) under patents now or hereafter owned or controlled by You, to Utilize the Modifications (or portions thereof), but solely to the extent that any such patent is reasonably necessary to enable Eclipse to Utilize the Modifications (or portions thereof), and not

to any greater extent that may be necessary to Utilize further Modifications or combinations.

3. DISTRIBUTION OBLIGATIONS

3.1. Application of License to You.
The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be distributed only under the terms of this License or a future version of this License released under Section 7.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.6.

3.2. Application of License to Eclipse.
(a) Other Products.
Eclipse may include Covered Code in products other than Genesis3D without such additional products becoming subject to the terms of this License, and may license such additional products on different terms from those contained in this License.
(b) Other Licenses.
Eclipse may license the Source Code including Modifications incorporated therein, without such additional products becoming subject to the terms of this License, and may license such additional products on different terms from those contained in this License.
(c) Other Releases.
Eclipse may create new releases of Genesis3D without such releases becoming subject to the terms of this License.

3.3. Availability of Source Code.
Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party. Furthermore, the Source Code to the Modifications You create or contribute must be delivered in fully compilable form using commonly available tools. If the tools used to compile the Source Code are not commonly available You must also include the tools, in Source Code and Executable forms, as part of the Modifications. Your Source Code must be delivered unencrypted, uncloaked, unobscured and in human readable form.

3.4. Description of Modifications.
You must cause all Covered Code to which you contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by Eclipse in (a) the Source Code, and (b) in any notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

3.5. Intellectual Property Matters
(a) Third Party Claims.
If You have knowledge that a party claims an intellectual property right in particular functionality or code (or its utilization under this License), you must include a text file with the source code distribution titled ``LEGAL" which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If you obtain such knowledge after You make Your Modification available as described in Section 3.3, You shall promptly modify the LEGAL file in all copies You make available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.
(b) Contributor APIs.
If Your Modification is an application programming interface and You own or control patents which are reasonably necessary to implement that API, you must also include this information in the LEGAL file.

3.6. Required Notices.
You must duplicate the notice in Exhibit A in each file of the Source Code, and this License in any documentation for the Source Code, where You describe recipients' rights relating to Covered Code. If You created one or more Modification(s), You may add your name as a Contributor to the notice described in Exhibit A. If it is not possible to put such notice in a particular Source Code file due to its structure, then you must include such notice in a location (such as a relevant directory file) where a user would be likely to look for such a notice. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of Eclipse or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify Eclipse and every Contributor for any liability incurred by Eclipse or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.7. Distribution of Executable Versions.
You may distribute Covered Code in Executable form only if the requirements of Section 3.1-3.6 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.3. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients' rights relating to the Covered Code. You may distribute the Executable version of Covered Code under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable version does not attempt to limit or alter the recipient's rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by Eclipse or any Contributor. You hereby agree to indemnify Eclipse and every Contributor for any liability incurred by Eclipse or such Contributor as a result of any such terms You offer.

3.8. Larger Works.
You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. RESTRICTIONS

4.1. You agree to:
(a) display the original, unmodified Genesis3D animated logo as the first logo on startup of your product, demo or application;
(b) prominently display the Genesis3D logo on any marketing materials, advertising or packaging of your product, demo or application;
(c) distribute the Covered Code to third parties who agree to be bound by these terms and conditions;
(d) make reasonable efforts to discontinue distribution of the Covered Code upon Eclipse's release of an update, upgrade or new version of the Covered Code and to make reasonable efforts to distribute such updates, upgrades or new versions to your customers who have received the Covered Code herein;
(e) be solely responsible for any update or support obligation or other liability which may arise from your distribution of the Covered Code.

4.2. You may not:
(a) remove, alter, obscure, or modify in any way the appearance or operation of the Genesis3D logo;
(b) copy the Covered Code, in whole or in part, except as provided in this License;
(c) make any statement that your product is "certified," or that its performance is guaranteed, by Eclipse.

5. INABILITY TO COMPLY DUE TO STATUTE OR REGULATION

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute or regulation then You must: (a) comply with the terms of this

License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.5 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

6. APPLICATION OF THIS LICENSE

This License applies to code to which Eclipse has attached the notice in Exhibit A, and to related Covered Code.

7. VERSIONS OF THE LICENSE

7.1. New Versions.
Eclipse may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.

7.2. Effect of New Versions.
Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Eclipse. No one other than Eclipse has the right to modify the terms applicable to Covered Code created under this License.

8. DISCLAIMER OF WARRANTY

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN ``AS IS'' BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT ECLIPSE OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

9. TERMINATION

This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

10. LIMITATION OF LIABILITY

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL ECLIPSE, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THAT EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

INDEMNIFICATION: YOU SHALL INDEMNIFY, HOLD HARMLESS, AND DEFEND ECLIPSE AND IT SUPPLIERS FROM AND AGAINST ANY CLAIMS OR LAWSUITS, INCLUDING ATTORNEY'S FEES, THAT ARISE OR RESULT FROM YOUR USE OR DISTRIBUTION OF ANY PRODUCT INCORPORATING THE COVERED CODE IN WHOLE OR IN PART.

11. U.S. GOVERNMENT END USERS

The Covered Code is a ``commercial item,'' as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of ``commercial computer software'' and ``commercial computer software documentation,'' as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

12. MISCELLANEOUS

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License is governed by the laws of the State of Texas and the United States, including patent and copyright laws. Any claim arising out of this License will be brought in Travis County, Texas.

The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

13. RESPONSIBILITY FOR CLAIMS

Except in cases where another Contributor has failed to comply with Section 3.5, You are responsible for damages arising, directly or indirectly, out of Your utilization of rights under this License, based on the number of copies of Covered Code you made available, the revenues you received from utilizing such rights, and other relevant factors. You agree to work with affected parties to distribute responsibility on an equitable basis.

EXHIBIT A.

``The contents of this file are subject to the Genesis3D Public License Version 1.01 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.genesis3d.com
Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.
The Original Code is Genesis3D, released March 25, 1999.
Copyright (C) 1996-1999 Eclipse Entertainment, L.L.C. All Rights Reserved.
Contributor(s): _____."

EXHIBIT B.

Functionality of the Covered Code includes:

(a) Representation and rendering of 2D and 3D images, objects, characters and scenes;
(b) Collision detection;
(c) Artificial Intelligence (AI), including path finding, planning and movement;
(d) Network and internet communications;
(e) Data compression;
(f) Streaming data over a network, modem, or the internet;
(g) Client/server architecture;
(h) Physics simulation;
(i) Creation and editing of world geometry;
(j) Character animation, including motion interpolation, motion planning, collision, and inverse kinematics.

This software uses the Bass Freeware sound library.

See http://www.un4seen.com/ for details.
Bass is used under the Shareware licence.

Copyright, Disclaimer, and all that other jazz
==========

The BASS library is free for non-money making use... if you are not charging for the software, then you can use BASS in it for free (a mention in the credits would be nice though!). If you wish to use BASS in shareware or commercial products, then see the next section.

This software is provided "as is", without warranty of ANY KIND, either expressed or implied, including but not limited to the implied warranties of merchantability and/or fitness for a particular purpose. The author shall NOT be held liable for ANY damage to you, your computer, or to anyone or anything else, that may result from its use, or misuse. Basically, you use it at YOUR OWN RISK.

Usage of BASS indicates that you agree to the above conditions.

You may freely distribute the BASS package as long as NO FEE is charged and all the files remain INTACT AND UNMODIFIED.

All trademarks and other registered names contained in the BASS package are the property of their respective owners.

BASS in shareware and commercial software?
==========

BASS is available for use in your shareware and commercial software. The license types available are as follows:

SHAREWARE: Allows the usage of BASS in an unlimited number of your shareware products.

SINGLE COMMERCIAL: Allows the usage of BASS in a single commercial product.

UNLIMITED COMMERCIAL: Allows the usage of BASS in an unlimited number of your commercial products. This license is on a per site basis. So if your company has two sites and both use BASS, then two licenses are required.

If you're an individual selling your own software, then the SHAREWARE license is the one for you. If you're a commercial entity (or your software is published/sold by one), then a COMMERCIAL license is required.

In all cases there are no royalties to pay, and you can use all future BASS updates without further cost. One license covers one person or entity and is not transferable.

These licenses do not allow reselling of BASS, ie. as part of another development library. If none of these licenses match your requirements, or if you have any questions about using BASS in shareware/commercial software, then please get in touch (email: bass@un4seen.com).

This software uses the FreeImage open source image library.
See http://freeimage.sourceforge.net for details.
FreeImage is used under the (GNU GPL or FIPL), version (licence version).

FreeImage Public License - Version 1.0

---

1. Definitions.

1.1. "Contributor" means each entity that creates or contributes to the creation of Modifications.

1.2. "Contributor Version" means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.

1.3. "Covered Code" means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof.

1.4. "Electronic Distribution Mechanism" means a mechanism generally accepted in the software development community for the electronic transfer of data.

1.5. "Executable" means Covered Code in any form other than Source Code.

1.6. "Initial Developer" means the individual or entity identified as the Initial Developer in the Source Code notice required by Exhibit A.

1.7. "Larger Work" means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.

1.8. "License" means this document.

1.9. "Modifications" means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a
Modification is:

A. Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.

B. Any new file that contains any part of the Original Code or previous Modifications.

1.10. "Original Code" means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.

1.11. "Source Code" means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated interface definition files, scripts used to control
compilation and installation of an Executable, or a list of source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor's choice. The Source Code can be in a compressed or archival form, provided the appropriate decompression or de-archiving software is widely available for no charge.

1.12. "You" means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of fifty percent (50%) or more of the outstanding shares or beneficial ownership of such entity.

2. Source Code License.

2.1. The Initial Developer Grant.
The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

(a) to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, or as part of a Larger Work; and

(b) under patents now or hereafter owned or controlled by Initial Developer, to make, have made, use and sell ("Utilize") the Original Code (or portions thereof), but solely to the extent that
any such patent is reasonably necessary to enable You to Utilize the Original Code (or portions thereof) and not to any greater extent that may be necessary to Utilize further Modifications or
combinations.

2.2. Contributor Grant.
Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

(a) to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code or as part of a Larger Work; and

(b) under patents now or hereafter owned or controlled by Contributor, to Utilize the Contributor Version (or portions thereof), but solely to the extent that any

such patent is reasonably necessary to enable You to Utilize the Contributor Version (or portions thereof), and not to any greater extent that
may be necessary to Utilize further Modifications or combinations.

3. Distribution Obligations.

3.1. Application of License.
The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or
restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

3.2. Availability of Source Code.
Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.

3.3. Description of Modifications.
You must cause all Covered Code to which you contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in (a) the Source Code, and (b) in any notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

3.4. Intellectual Property Matters

(a) Third Party Claims.
If You have knowledge that a party claims an intellectual property right in particular functionality or code (or its utilization under this License), you must include a text file with the source code distribution titled "LEGAL" which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If you obtain such knowledge after You make Your Modification available as described in Section 3.2, You shall promptly modify the LEGAL file in all copies You make
available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.

(b) Contributor APIs.
If Your Modification is an application programming interface and You own or control patents which are reasonably necessary to implement that API, you must also include this information in the LEGAL file.

3.5. Required Notices.
You must duplicate the notice in Exhibit A in each file of the Source Code, and this License in any documentation for the Source Code, where You describe recipients' rights relating to Covered Code. If You created one or more Modification(s), You may add your name as a Contributor to the notice described in Exhibit A. If it is not possible to put such notice in a particular Source Code file due to its
structure, then you must include such notice in a location (such as a relevant directory file) where a user would be likely to look for such a notice. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear than any such warranty, support, indemnity or
liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of
warranty, support, indemnity or liability terms You offer.

3.6. Distribution of Executable Versions.
You may distribute Covered Code in Executable form only if the requirements of Section 3.1-3.5 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.2. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You
describe recipients' rights relating to the Covered Code. You may distribute the Executable version of Covered Code under a license of Your choice, which may contain terms different from this License,
provided that You are in compliance with the terms of this License and that the license for the Executable version does not attempt to limit or alter the recipient's rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.7. Larger Works.
You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation.

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Application of this License.

This License applies to code to which the Initial Developer has attached the notice in Exhibit A, and to related Covered Code.

6. Versions of the License.

6.1. New Versions.
Floris van den Berg may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.

6.2. Effect of New Versions.
Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Floris van den Berg
No one other than Floris van den Berg has the right to modify the terms applicable to Covered Code created under this License.

6.3. Derivative Works.
If you create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), you must (a) rename Your license so that the phrases "FreeImage", `FreeImage Public License", "FIPL", or any confusingly similar phrase do not appear anywhere in your license and (b) otherwise make it clear that your version of the license contains terms which differ from the FreeImage Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in Exhibit A shall not of themselves be deemed to be modifications of this License.)

7. DISCLAIMER OF WARRANTY.

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

8. TERMINATION.

This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

9. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE
EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THAT EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

10. U.S. GOVERNMENT END USERS.

The Covered Code is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

11. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by Dutch law provisions (except to the extent applicable law, if any, provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in, the The Netherlands: (a) unless otherwise agreed in writing, all disputes relating to this License (excepting any dispute relating to intellectual property rights) shall be subject to final and binding arbitration, with the losing party paying all costs of arbitration; (b) any arbitration relating to this Agreement shall be held in Almelo, The Netherlands; and (c) any litigation relating to this Agreement shall be subject to the jurisdiction of the court of Almelo, The Netherlands with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys fees and expenses. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. RESPONSIBILITY FOR CLAIMS.

Except in cases where another Contributor has failed to comply with Section 3.4, You are responsible for damages arising, directly or indirectly, out of Your utilization of rights under this License, based
on the number of copies of Covered Code you made available, the revenues you received from utilizing such rights, and other relevant factors. You agree to work with affected parties to distribute
responsibility on an equitable basis.

EXHIBIT A.

"The contents of this file are subject to the FreeImage Public License Version 1.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://home.wxs.nl/~flvdberg/freeimage-license.txt

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

****

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN
OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS
TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE
PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,
REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY
YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

VERSION ORIGINAL

El manual original lo podeis conseguir en la pagina de entidad 3d " www.entidad3d.com" o en el siguiente link.
http://www.entidad-3d.com/tutorial_entidad_3d/tutorial_entidad_3d.zip

REVISION DE ESTE MANUAL:

Este manual es una versión no oficial imprimible del original de ENTIDAD3D, realizada por Jaime Yemaíl.